Universidad
Carlos III de Madrid

Proyecto Fin de Carrera

Diseño de un driver de disco duro para sistemas de ficheros indetectables

06.09.2015

—

Autor: Daniel O'Grady Rueda
Tutor: Víctor Suárez Paniagua
Ingeniería Técnica en Telecomunicaciones
Especialidad Sonido e Imagen
Universidad Carlos III
Leganés, Madrid

Página dejada intencionalmente en blanco

Título	Diseño de un driver de disco duro para sistemas de ficheros indetectables
Autor	Daniel O'Grady Rueda
Tutor	Víctor Suárez Paniagua

EL TRIBUNAL

Presidente	Pedro Peris-López
Vocal	Mª Celeste Campo Vázquez
Secretario	David Expósito Singh

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 27 de octubre de 2015 en Leganés, Madrid, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

A Víctor, por ayudarme y guiarme en la resolución de este proyecto;

A mis padres, por su inconmensurable paciencia y apoyo;

A Almudena, por estar a mi lado todo este largo tiempo;

Y a mis socios y resto de amigos, que nunca han dejado de apoyarme.

Resumen

Se desarrollará un controlador de disco duro, especialmente diseñado para funcionar bajo Linux con una serie de discos duros presentes en el mercado actual, capaz **escribir** en y **leer** de las **áreas reservadas del disco duro** y de presentar el espacio disponible al sistema operativo como un dispositivo de bloques **formateable** y usable como cualquier otra unidad de disco. El desarrollo se enmarca dentro del ámbito de la **informática forense**, facilitando y ampliando las capacidades de investigación del analista forense, y se hará en el lenguaje de programación C con las cabeceras adecuadas de Linux, teniendo como objetivo ser funcional para versiones del **kernel de Linux** compatibles con la **2.6**, en cuanto a la capa de dispositivos de bloque se refiere.

Palabras clave

Disco duro, *driver*, áreas reservadas, informática forense, ocultación de información.

Abstract

The aim of this project is to develop a hard disk driver, especially crafted to work under Linux with a series of hard disk drives very present in the market, which is able to **write to** and **read from** the **reserved areas of the disk**, and present the space available as block device to the operating system, which in turn will be able to **format** it and use it as any other block device registered on the system. This development is framed inside the scope of the **Computer Forensics** investigations, making it possible for the forensic examiner to access previously unreachable areas of the disk, and will be written in the C programming language, with the appropriate Linux headers, with the aim of being fully functional under **Linux kernel** versions compatible with —regarding the block device layer— version **2.6**.

Keywords

Hard disk, driver, reserved areas, computer forensics, information hiding.

Índice general

Contenido

Índice de figuras

Índice de tablas

Índice de fórmulas

Glosario de términos

1. **AAM**: *Automatic Acoustic Management*: Sistema que posee opcionalmente un disco duro para el control automático del ruido que emiten (a costa de variar velocidades de búsqueda, etc.).
2. **ABA**: *Absolute Block Addressing*, direccionamiento absoluto por bloques. Utilizado en traducción de direcciones físicas a lógicas en sistemas de almacenamiento para referirse de forma lineal a los sectores o unidades mínimas de almacenamiento, introduciendo en el cálculo de las direcciones los sectores descartados por otros sistemas de direccionamiento como LBA (función biyectiva *sector físico* $\leftrightharpoons$ *ABA*).
3. **APM**: *Automatic Power Management*: Sistema que posee opcionalmente un disco duro para el control automático de la energía que consumen.
4. **ASCII**: *American Standard Code for Information Interchange*, es un esquema de codificación de caracteres utilizado para representar texto en sistemas computadores, actualmente su uso está cayendo en favor de otros esquemas más amplios y modernos.
5. **ATA**: *Advanced Technology Attachment*, estándar de interconexión de dispositivos de almacenamiento con otros sistemas, que derivó en los estándares P-ATA y el actual S-ATA.
6. **ATA/ATAPI**: *AT Attachment with Packet Interface*, protocolo que especifica los requisitos de compatibilidad (el interfaz) para la interconexión de dispositivos de almacenamiento y otros sistemas. Es una generalización del estándar ATA para dispositivos diferentes a los discos duros.
7. **BIOS**: *Basic Input Output System*, sistema básico de entrada y salida en un sistema computador, encargado de realizar las gestiones y comunicaciones con periféricos a bajo nivel y del arranque del sistema.
8. **Cabezal**: En un disco duro, dispositivo electrónico encargado de la lectura y/o escritura de los dominios magnéticos en que se almacena la información.

9. **CHS**: *Cylinder-Head-Sector*, sistema de direccionamiento en sistemas de almacenamiento que no abstrae las particularidades físicas del mismo (identifica un sector único haciendo referencia al cilindro, cabezal de ese cilindro y sector de ese cabezal), usado antiguamente en discos duros y otros dispositivos.
10. **Cilindro**: En un disco duro con varias superficies activas (con uno o varios platos magnéticos), conjunto de pistas con el mismo radio.
11. **CPU**: *Central Processing Unit*, circuito integrado que es la principal unidad de proceso de datos en un sistema computador. Usado en este artículo para referirse de forma general a cualquier circuito integrado microcontrolador.
12. **DCO**: *Device Configuration Overlay*, un área del disco duro (diferente del área de sistema) que puede ocultarse al sistema operativo; la mayoría de herramientas forenses en el mercado[1] son conscientes de la posible existencia de estas zonas y el estándar ATA/ATAPI-6 provee mecanismos para identificar y eliminar estas zonas.
13. **ECC**: *Error Correcting Code*, código de corrección de errores. Es un mecanismo que se utiliza en sistemas de recuperación de información (análisis y detección de la señal) para detectar y corregir, mediante la inclusión de datos redundantes en el flujo de información, errores que pudieran detectarse en la reconstrucción de la información original. En tecnología de almacenamiento de datos, cada sector posee un campo de ECC que ayuda al software del dispositivo a detectar problemas en la lectura, y corregir fallos menores.
14. **EEPROM**: *Electronically Erasable Programmable Read Only Memory*, circuito integrado de memoria de sólo lectura que puede ser borrado y reprogramado electrónicamente.

[1] Se puede encontrar una variada oferta en Internet de herramientas que anuncian soporte para detectar estas zonas, como la herramienta *ZClone Xi* de Logicube: http://www.logicube.com/wp-content/uploads/2015/02/ZCloneXi-Data-Sheet-v7.pdf

15. **E/S**: Entrada y salida. Utilizado para referirse a dispositivos que admiten flujos bidireccionales de información (lectura y escritura), como los dispositivos de almacenamiento.
16. **Firmware**: Programa informático o software que proporciona control a bajo nivel de diversos sistemas informáticos. Usado en este documento para referirse al software propietario que gobierna el funcionamiento interno de un disco duro.
17. **Flash**: Tipo de memoria permanente que puede ser borrada y reprogramada eléctricamente. Usado en este artículo para referirse a las unidades de memoria permanente o ROM instaladas habitualmente en discos duros modernos, donde se almacena parte del firmware del mismo.
18. **GMR**: *Giant Magnetoresistance*, efecto mecanocuántico usado en los sensores de lectura en discos duros modernos.
19. **GNU**: *GNU's not Unix!*, sigla recursiva para referirse a una extensa colección de software gratuito usada para construir un sistema operativo de tipo UNIX.
20. **GPL**: *GNU Public License*, licencia de software gratuito que garantiza al usuario final el derecho de usar, estudiar, modificar, copiar y redistribuir el software, siempre bajo los mismos términos de licencia.
21. **GPU**: *Graphics Processing Unit*, circuito integrado responsable del procesado de datos gráficos.
22. **H.M., *Head Map* o Mapa de cabezales**: Es un término que se utiliza para identificar los cabezales de lectura/escritura activos en un disco duro (frente a los instalados, o para referirse a ellos).
23. **HPA**: *Host Protected Area*, zona del disco duro que se puede ocultar al sistema operativo, similar a la DCO, a diferencia de que la HPA puede desactivarse temporalmente (para un ciclo de encendido) en la memoria RAM del disco duro.

24. **LBA**: *Logical Block Addressing*, sistema de direccionamiento usado en dispositivos de almacenamiento que atribuye un único número, en secuencia creciente y partiendo desde 0, para cada sector usable del dispositivo.
25. **LSW**: *Least Significant Word*, palabra o conjunto de 2 bytes menos significativo en un bloque de datos. Representa los 16 bits con menor peso en una cadena de bits.
26. **MSW**: *Most Significant Word*, palabra o conjunto de 2 bytes más significativo en un bloque de datos. Representa los 16 bits con mayor peso en una cadena de bits.
27. **NVRAM**: *Non-Volatile RAM*, nombre genérico para referirse a memorias de acceso aleatorio de tipo ROM.
28. **P-ATA**: *Parallel ATA*, interfaz de comunicaciones ATA en paralelo, usado para interconectar dispositivos de almacenamiento con otros sistemas, obsoleto en la actualidad.
29. **Padding**: Usado para hacer referencia a los datos que se usan como relleno en una cadena de información que, por algún motivo, necesita tener una longitud determinada pero la información útil de la que se dispone es de menor longitud que la necesaria. Habitualmente se usan bytes nulos como relleno (0x00).
30. **Pista**: Cada una de las circunferencias concéntricas de una superficie de un plato de un disco duro, en la que se escribe información agrupada en sectores.
31. **RAM**: *Random Access Memory*, circuito integrado que es la principal fuente de memoria volátil de acceso aleatorio en un sistema computador. Usado en este artículo para referirse de forma general a la información contenida en la memoria volátil.
32. **RPM**: *Revolutions Per Minute*, unidad de medida para medir frecuencia, revoluciones o ciclos por minuto de un motor o un sistema pulsante.

33. **ROM**: *Read Only Memory*, circuito integrado de memoria de sólo lectura. Utilizado en ocasiones para referirse genéricamente a otros tipos de memorias de sólo lectura.
34. **S.A. o *Service Area***: Área de sistema o área reservada de un disco duro, donde se almacena parte del código de funcionamiento además otra información de arranque y monitorización del disco.
35. **S-ATA**: *Serial ATA*, interfaz de comunicaciones ATA en serie, usado para interconectar dispositivos de almacenamiento con otros sistemas, usado en la actualidad y sucesor del antiguo P-ATA.
36. **Sector**: Unidad mínima de lectura/escritura en un disco duro o dispositivo de almacenamiento. Su tamaño físico —que no el lógico— no está fijado por ningún estándar, pero la industria ha adoptado el valor de 512 bytes como un estándar de facto; desde 2010 se empiezan a ver también sectores de 4096 kB.
37. **SMART o S.M.A.R.T.**: *Self-Monitoring, Analysis and Reporting Technology*, es un sistema de monitorizado opcional incluido en la mayoría de discos duros del mercado y definido en el estándar ATA/ATAPI-5.
38. **SMR**: *Shingled Magnetic Recording*, sistema de almacenamiento utilizado en algunos discos duros modernos diseñados para *almacenamiento en frío* o a largo plazo, que solapa parcialmente las pistas o sectores durante la escritura para aumentar la densidad de información.
39. **SPT**: *Sectors Per Track*, sectores en un disco duro por cada pista del mismo. Es una unidad de densidad de datos en un dispositivo de almacenamiento de revolución.
40. **UNIX**: Marca comercial del sistema operativo Unix.
41. **VSC**: *Vendor Specific Command*, comando específico del fabricante. Usado para referirse a los comandos ATA no estándar, que cada fabricante se reserva para comunicarse con sus discos duros para realizar diversas

operaciones, como actualizaciones de firmware, consulta de registros o *logs*, o reparaciones o modificaciones.

Capítulo 1

Introducción y objetivos

1.1. Introducción

El disco duro ha sido probablemente el único dispositivo electrónico que sobrevive en la actualidad y en mayor medida a las predicciones de la ley de Moore (o, específicamente, la ley de Kryder): a medida que nos hemos ido adentrando en la segunda década del siglo XXI, dispositivos electrónicos como las CPU, las GPU, o las memorias volátiles RAM han ido estancándose en prestaciones, encontrándose con grandes problemas técnicos y físicos difíciles de superar sin un cambio fundamental en las tecnologías estrella de la computación actual, mientras que las unidades de disco duro y almacenamiento han sabido adaptarse a cambios tecnológicos de gran calado, sin un cambio en el formato en que éstos se presentan al consumidor, ni un cambio mayor en la forma en que interactúan con el resto de dispositivos. Este hecho ha supuesto que hoy en día, en cada hogar de cualquier país desarrollado, existan ingentes cantidades de información almacenada en decenas de diferentes dispositivos, desde ordenadores portátiles, de escritorio, unidades ópticas de diferentes tipos, memorias *flash*, teléfonos móviles y *tablets*, y otros dispositivos, a la vez que la capacidad para procesar toda esta información no ha crecido sustancialmente, y mucho menos a la par.

Esto supone un gran y creciente problema en el ámbito de las tecnologías de la información para diversos actores; entre ellos, las figuras del *especialista en evidencia digital* (DES, del inglés *Digital Evidence Specialist*), y del *interventor de evidencia digital en primera instancia* (DEFR, del inglés *Digital Evidence First Responder*), ambas enmarcadas en el área de la informática forense. Estas figuras,

que son las responsables de obtener y analizar los repositorios de información durante investigaciones que involucran dispositivos informáticos en lo que se denomina la *investigación informático forense*, han visto incrementada enormemente en los últimos años la cantidad de tiempo que han de dedicar para obtener y procesar la información recolectada durante la investigación.

Consecuencia probable del aumento en la carga de trabajo de estas figuras, y de la necesidad de definir unas líneas generales de actuación hasta hace poco inexistentes, en 2012 vio al fin la luz el estándar universal BS ISO/IEC 27037:2012, que ha unificado diferentes estándares y costumbres locales usados para actuar en estos casos. El estándar es cuidadoso a la hora de elegir las palabras y formular y definir las líneas generales de actuación del investigador forense; no obstante la ambigüedad que lo caracteriza no es suficiente para ocultar una falta de completitud a la hora de contemplar los diferentes escenarios que se le pueden presentar al examinador, especialmente en cuanto a la definición de *medios digitales de almacenamiento* que de él se extrae, y al carácter de suficiencia del que dota a las tareas técnicas realizadas con *herramientas validadas del sector*.

1.2. Objetivos

El objetivo del presente proyecto es estudiar el carácter de suficiencia o insuficiencia tanto de la definición de *medios digitales de almacenamiento* que del estándar se extrae, como de la validez de apoyar la investigación únicamente en *herramientas validadas del sector*, en lugar de un enfoque más amplio. Para ello, se propone el desarrollo de un controlador o *driver* de disco duro capaz de ocultar cierta cantidad de información útil en un disco duro convencional, pasando la misma completamente desapercibida a ojos de las *herramientas validadas del sector*, por quedar directamente fuera de su alcance. El proyecto pretende así mismo ampliar el alcance de la definición de *medios digitales de almacenamiento*, así como fomentar el uso de

herramientas de este tipo, no validadas por el sector, que pueden ayudar a no pasar por alto información de otro modo invisible para el investigador.

El desarrollo del driver se hará en el lenguaje de programación C y para Linux. El código actualizado será liberado mediante una licencia Creative Commons para que pueda ser *portado* a otras plataformas y compilado para diferentes arquitecturas. Como entregable anexo al proyecto, se dispondrá el código completo del driver, así como una prueba de su funcionamiento.

1.3. Esquema de esta memoria

La memoria del presente proyecto de investigación y desarrolla se estructura en los siguientes bloques:

1.3.1. Bloque introductorio

a. Introducción y objetivos, donde se establecen los antecedentes (**Capítulo 1**).
b. Motivaciones y estado del arte, donde se analiza el panorama actual, soluciones actuales al problema y deficiencias en las formas actuales (**Capítulo 2**).

1.3.2. Bloque de análisis

c. Fundamentos teóricos y características de funcionamiento interno de los discos duros actuales. Análisis de soluciones existentes (**Capítulo 3**).

1.3.3. Bloque de desarrollo

d. Desarrollo del controlador del disco duro en sus distintas fases (**Capítulo 4**):
    i. Desarrollo de una técnica de acceso a las áreas reservadas del disco duro.
    ii. Desarrollo de un controlador de dispositivo de bloques básico.
    iii. Integración de la técnica de acceso a las áreas reservadas en el controlador básico.

iv. Desarrollo de un traductor de direcciones unidimensionales (LBA) en tridimensionales (CHS).

1.3.4. Bloque de gestión

e. Planificación y presupuesto (**Capítulo 5**).

1.3.5. Corolario

f. Conclusiones (**Capítulo 6**).

g. Líneas futuras de desarrollo (**Capítulo 7**).

1.4. Medios de los que se ha dispuesto

Para llevar a cabo el desarrollo de este controlador, se ha dispuesto de una serie de elementos tanto software como hardware, que se listan a continuación:

1.4.1. Elementos hardware

h. Ordenador de sobremesa con los elementos básicos.

i. Controladora especial de disco duro PC3000 de ACE Laboratory.

j. Un set de 3 discos duros de la marca Western Digital, para hacer pruebas.

1.4.2. Elementos software

a. Sistema operativo Linux Debian 7 i686.

b. Toda la suite de compilación de módulos para el kernel de Linux y las cabeceras del kernel 2.6 presente en la máquina.

c. Sistema operativo Windows 7 de 32 bit.

d. Software PC3000 Utility, para comunicaciones con el disco duro sin intervención del BIOS del ordenador, y con acceso directo a los registros ATA.

Capítulo 2

Motivaciones y estado del arte

2.1. Motivaciones

Toda investigación informático forense tiene un proceso a seguir, que comienza con la personificación del examinador o experto forense en el lugar donde se han de recolectar las evidencias, y culmina con la elaboración de un informe de conclusiones a las que se ha llegado durante la investigación, posiblemente para su defensa ante un juzgado. El proceso, según marca el estándar ISO/IEC 27037, se basa en los principios de auditabilidad, justificabilidad, repetibilidad y reproducibilidad. Para garantizar estos cuatro principios sobre los que se fundamenta la norma, se establecen una serie de pautas, siendo las más representativas la ejecución de la investigación en un entorno controlado y *limpio* en términos forenses, la elaboración y el cumplimiento con una cadena de custodia, la correcta etiquetación de cada pieza incautada para su posterior investigación, y el uso de herramientas validadas por el sector. Pese a no haber dispuesto de una norma común en la gestión de evidencias digitales hasta la publicación de este estándar en octubre de 2012, los investigadores forenses disponían ya de diferentes códigos de buenas prácticas muy en línea con el estándar, como el NCJ 217679 utilizado por el servicio secreto estadounidense [1], y las herramientas comerciales más importantes han sido utilizadas como referente en la redacción del mismo, siendo prueba de ello el hecho de que no han visto necesario sufrir modificaciones severas tras su publicación.

Tras varios años trabajando en el ámbito de la informática forense, y tras haber trabajado con y recibido formación en los productos estrella del mercado para el análisis forense (Guidance Software EnCase® Forensic, AccessData FTK, entre otras), he podido comprobar en el desarrollo de mi labor profesional cómo, en muchos casos,

los investigadores gastamos gran parte del tiempo preparando y procesando las evidencias recolectadas para agilizar el acceso y la búsqueda de información relevante, ciñéndonos a lo que las herramientas nos proporcionan, ya que las mismas prometen no dejar ni un solo recoveco de las evidencias digitales por analizar.

Paralelamente, algunas de las investigaciones forenses desarrolladas involucraban evidencias electrónicas dañadas o en muy mal estado, que había que reparar previo al comienzo de la investigación. En el caso de los discos duros, las reparaciones pasaban muchas de las veces por realizar modificaciones en el *firmware* del mismo (el programa interno que los hace funcionar), además de la sustitución de algunos de los componentes. Esta tarea se realizaba en el laboratorio de recuperación de datos, donde pude ahondar en mis conocimientos sobre el funcionamiento interno de los discos duros, y sobre las características y peculiaridades de cada familia y fabricante de discos duros.

Las tareas de modificación del firmware de los discos son las que más conocimientos proporcionaban a los técnicos, ya que cada disco duro requiere sus técnicas particulares para acceder a unas zonas específicas, donde se puede modificar el comportamiento del disco duro en caso de detectar errores o problemas de funcionamiento. Estas zonas del disco duro, a las que había que acceder en caso de encontrar problemas de lectura para tratar de circunvenirlos, eran absolutamente obviadas en un análisis forense corriente, y gracias al tiempo dedicado al laboratorio de recuperación de datos pude comprobar que:

a) Esta zona del disco es una zona reservada del mismo (es decir, no accesible normalmente),

b) Esta zona es susceptible de realizar sobre ella operaciones tanto de lectura como de escritura en muchos modelos de disco duro, y

c) Esta zona reservada puede llegar a tener un tamaño nada insignificante, en la cual se puede guardar suficiente cantidad de información sensible como para convertirse en una preocupación mayor [2].

Tras muchas búsquedas y consultas con otros investigadores, llegué a la conclusión de que efectivamente se trataba de un territorio inexplorado en el mundo forense, y que cualquier información que pudiese haber escondida en esta zona quedaba directamente descartada en la investigación porque estaba directamente fuera del ámbito de cualquier investigación forense tradicional sobre un disco duro, dado que ninguna de las herramientas validadas por el sector está preparada para acceder directamente a estas zonas e incluirlas dentro del ámbito de la investigación.

Durante la investigación realizada sobre las características de estas zonas reservadas, pude comprobar que el problema puede ser realmente difícil de solucionar sin ayuda de los fabricantes: el problema no se queda en las herramientas, sino que para el acceso a las áreas reservadas del disco no hay estándar definido, ni consenso entre los fabricantes, ni controladores públicos conocidos para trabajar con estas zonas, y no todos los fabricantes están dispuestos a revelar detalles sobre el funcionamiento interno de sus dispositivos.

El presente proyecto tiene por objetivo último servir a la comunidad forense para ampliar el alcance de sus investigaciones un paso más, ofreciendo una herramienta que dé acceso a estas zonas restringidas de una forma útil, rápida y sencilla, y servir como punto de partida para desarrollar controladores más rápidos, mejores, y compatibles con más discos duros del mercado.

2.2. Estado del arte

Es sabido en la actualidad, con mayor o menor profundidad, acerca del funcionamiento interno de un disco duro: forma parte de la cultura digital el hecho de que esté compuesto por unos platos magnéticos, unos cabezales lectores, un motor y una placa electrónica (ver Figura 1).

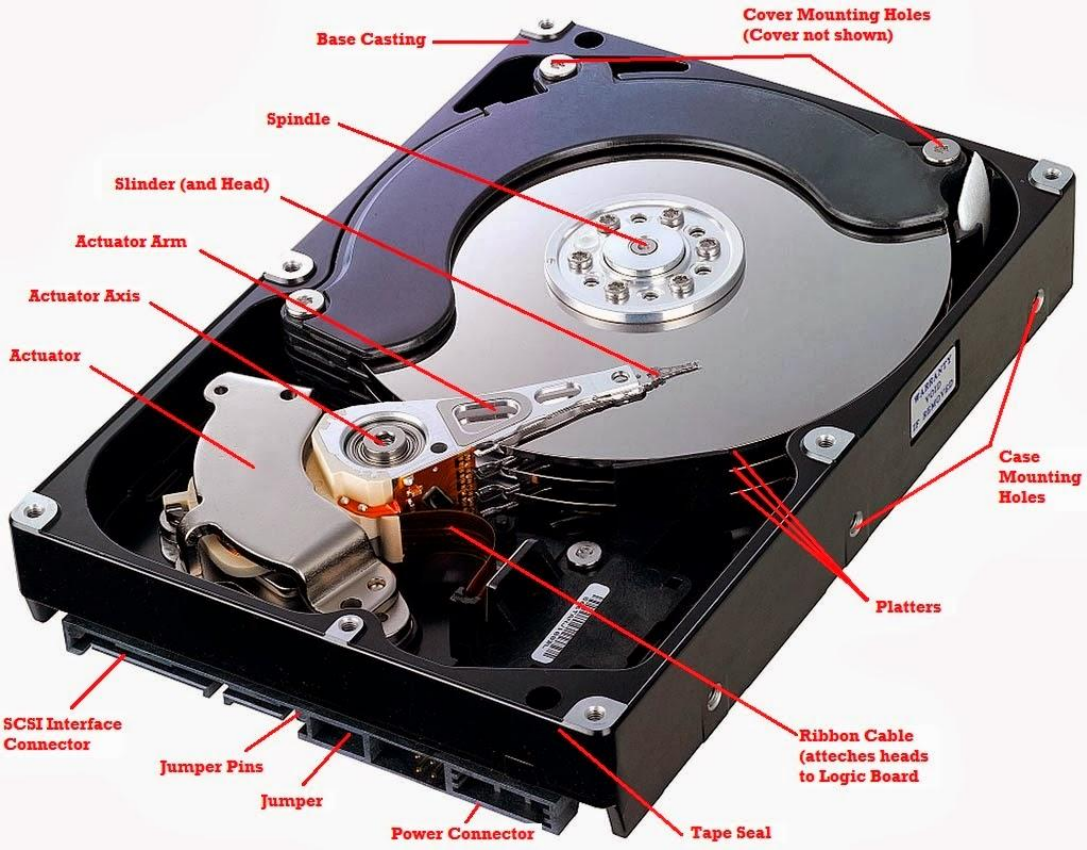

Figura 1: Bsierad. *Assembling Process and Function HDD hard drive parts*. Extraído de <http://www.bsierad.com/assembling-process-and-function-hdd-hard-drive-parts/>

En cualquier sistema operativo moderno que ofrece acceso a unidades de disco existe alguna implementación del protocolo ATA/ATAPI conforme a los requisitos del mismo [3], al que se ciñen los principales fabricantes de unidades de disco duro magnético presentes en el mercado actual (tras la desaparición formal de Fujitsu, Maxtor y Quantum): Western Digital, Seagate, y Toshiba (ver Figura 2).

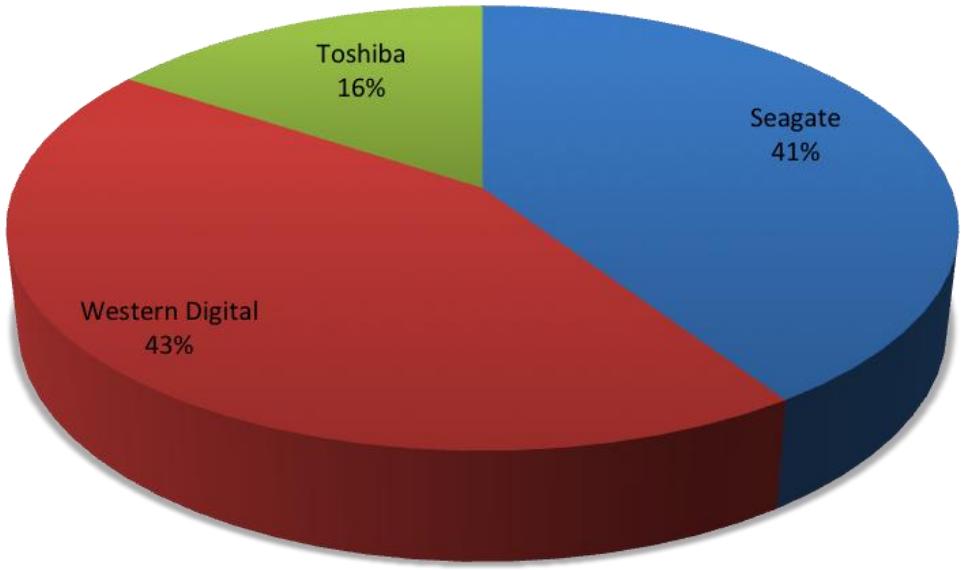

Figura 2: Tom Coughlin. Forbes. *HDD Annual Unit Shipments Increase In 2014*. Extraído de <http://www.forbes.com/sites/tomcoughlin/2015/01/29/hdd-annual-unit-shipments-increase-in-2014/>

La estructura interna de los discos (platos magnéticos, motor, cabezales, electrónica) es conocida por su aparente simpleza y la disponibilidad de especificaciones técnicas de los fabricantes [4], en las que destacan avances y mejoras competitivas; la parte más conocida acerca de su funcionamiento es la que forma parte del documento público que es el protocolo vigente ATA/ATAPI-8. Lo que se escapa al conocimiento público es la implementación interna de cada uno de los mecanismos que propone el estándar, así como el impacto que tiene en el funcionamiento del dispositivo, especialmente cuando el mismo es sometido a condiciones de estrés o malfuncionamiento: el estándar dicta un mecanismo para informar sobre errores encontrados en cualquiera de las cuatro capas que identifica [5] (las capas física, de enlace, de transporte y de comando/aplicación), pero no detalla más que la forma de informar a las capas superiores, y finalmente al sistema huésped, sobre el error ocurrido y limitadas acciones a realizar (como un reinicio y el establecimiento de algunos bits de estado) [6]. Cada fabricante, posteriormente, es libre de implementar estas rutinas a su manera, y de diferenciarse del resto desarrollando técnicas que

además de comportarse según dicta el estándar, ofrezca una capa más de protección, transparente tanto al usuario, como al protocolo ATA/ATAPI. En el caso del comportamiento que deben exhibir frente a errores en la capa física, una vez reportado el problema, el fabricante es libre de tratar de corregir el problema internamente, sin mediación del protocolo, para evitar que se vuelva a producir. Un caso común es el encontrado durante la lectura de sectores defectuosos en discos duros: los discos modernos cuentan con una reserva de sectores en cada pista y en cada superficie activa, y el firmware del disco duro es libre de tratar de sustituir los sectores que han reportado algún error en lectura o escritura por sectores de la reserva.

El estándar tampoco dicta la forma en que estos dispositivos deben organizarse internamente, ni cómo debe ser o en qué lugar debe residir su memoria de programa. Todo lo que el estándar ofrece es una posibilidad para implementar un mecanismo de actualización del *microcódigo* o firmware del dispositivo mediante el comando DOWNLOAD MICROCODE [7], pero la manera en que es almacenado, accedido o modificado el firmware de los discos duros de forma directa y selectiva no forma parte del estándar, y cada fabricante utiliza las técnicas propietarias que crea más convenientes para almacenar esta información, así como para acceder a ella modificarla posteriormente y fuera de fábrica, si fuera necesario. Todo lo que podemos saber a priori acerca de dónde reside esta información (que llamaremos a lo largo del documento e indistintamente *zonas restringidas del disco duro*, *área de sistema* o *área de servicio*), o la manera en que se accede a ella, es que se trata de información privada, dependiente del fabricante, y no disponible públicamente.

En la actualidad existen herramientas comerciales especializadas que ofrecen acceso a las zonas restringidas del disco duro, todas ellas especialmente focalizadas en la reparación de los mismos, y en la recuperación de datos de discos fallidos que no se pueden reparar. Las más conocidas y establecidas de estas herramientas se listan a continuación, junto con el nombre de la empresa desarrolladora o distribuidora:

- PC3000, de ACE Laboratory
- Atola Insight, de Atola Technology
- MRT Data recovery, de MRT Lab
- HD Doctor, de Salvation Data Technology

Estas herramientas ofrecen la posibilidad, a un coste relativamente alto[2], de acceder a las zonas reservadas del disco duro usando hardware especializado y una *suite* de software propia y de código cerrado. Ninguna proporciona información sobre las técnicas que utilizan o los comandos mediante los cuales se comunican con el disco duro; todas ellas ofrecen una interfaz gráfica de usuario con botones y diversos paneles mediante los cuales se lanzan las comunicaciones con el disco.

Estas herramientas no se utilizan en forense principalmente por dos motivos:

1. No garantizan, al contrario que las herramientas validadas del sector, la integridad de los datos a los que acceden, ni proporcionan ningún mecanismo para ello: la integridad de los datos depende únicamente de la pericia del investigador.
2. Las técnicas que utilizan para acceder a las zonas restringidas de los discos duros no son conocidas ni los resultados obtenidos presentados de acuerdo a una regla o norma, por lo que los pasos seguidos pueden ser repetidos pero los resultados no pueden ser reproducidos (como exige la norma ISO/IEC 27037 [8]), ni validados.

Aún teniendo acceso a las zonas reservadas de los discos, estas herramientas están diseñadas para reparar discos o recuperar información de dispositivos dañados, y el acceso (lectura y escritura) a estas zonas se utiliza como medio para la reparación o recuperación de la información, pero nunca como fin en sí mismo: se accede a estas zonas para modificar diversos registros que afectan al funcionamiento del dispositivo, y para almacenar copias de seguridad de determinados bloques de información

[2] El coste de estas herramientas está sujeto a cambios según la zona en que se distribuye y actualmente ningún fabricante anuncia precios en su sitio web; se dispone de una unidad PC3000 de ACE Laboratory cuyo coste se detalla en el Capítulo 4.

críticos previo a la modificación de los mismos, pero no existe ningún tipo de garantía sobre la integridad de la información recabada, ni se conoce el mecanismo mediante el cual se ha extraído.

Es importante recalcar que todas las herramientas mencionadas cuentan con un hardware particular que hace de interfaz con el disco duro, y es preciso que el disco esté conectado a sus módulos especializados, en lugar de estar directamente conectado a los puertos ATA del ordenador en cuestión, como normalmente se haría (ya sea P-ATA o S-ATA) [9] [10] [11] [12].

Por otro lado también es importante recalcar que ninguna de las anteriores herramientas implementa la interfaz de dispositivo de bloques correspondiente (según sistema operativo), para ofrecer acceso a las zonas reservadas del disco duro mediante los mecanismos habituales (las llamadas *system calls*, o interrupciones de BIOS como INT 13h), sino que todas ellas usan un controlador propietario cuyo funcionamiento queda restringido al uso de la aplicación para la cual se ha desarrollado dicho controlador.

En este proyecto se propone el desarrollo de un controlador que sirva como interfaz entre el disco duro y el sistema operativo, que sea capaz de acceder a las zonas restringidas del disco duro y presentarlas al sistema operativo como un dispositivo más, trayéndolas al frente y convirtiéndolas automáticamente en un elemento más accesible al investigador forense, sujeto a adquisición y examen.

Capítulo 3

Fundamentos teóricos y características de funcionamiento de un disco duro actual

3.1. Componentes de un disco duro actual

Un disco duro común, como los que se pueden encontrar en cualquier computador familiar o en cualquier oficina, es un dispositivo de almacenamiento magnético compuesto de las siguientes partes [13] (ver Figura 1):

1. Uno o varios discos o platos magnéticos. Son discos cuya composición varía de un modelo de disco a otro, recubiertos en todo caso de una película de un material ferromagnético, en una o varias capas, muy finamente pulido. Es el medio en el que se almacena la información, mediante alguna técnica de grabado: GMR, SMR, etc.
2. Uno o varios cabezales de lectura/escritura. Cada superficie activa del disco duro (es decir, cada superficie magnética) es accedida mediante un cabezal de lectura/escritura, que flota sobre la superficie de los discos y es capaz de detectar pequeñas variaciones en los campos magnéticos grabados en la superficie de los discos, así como de inyectar pulsos magnéticos enfocados para grabar información sobre las superficies.
3. Un brazo actuador. Los cabezales van montados sobre el brazo actuador, que es accionado mediante un mecanismo de bobina de voz. La bobina controla de forma muy precisa la posición radial en la que se encuentran los cabezales variando el ángulo de giro del brazo actuador.
4. Un preamplificador de señal. La señal que leen los cabezales de las superficies magnéticas es extremadamente frágil y débil, y debe procesarse lo antes posible. En la actualidad, el preamplificador de señal va montado sobre el

mismo brazo actuador, minimizando la distancia a la que hay que llevar la señal sin amplificar y minimizando así la cantidad de ruido en las puertas del amplificador.

5. Un motor que asista en el giro de los discos magnéticos. Los discos duros de escritorio actuales giran a velocidades que oscilan entre las 5400 y las 7200 RPM; el giro ha de ser muy estable y gracias a las velocidades alcanzadas se crea una fina capa de aire con el giro que sirve de *pista de patinaje* a los cabezales de lectura y escritura.
6. Un servomecanismo de control que autorregule la velocidad de giro de los platos magnéticos y corrija las pequeñas desviaciones en el trazado de las pistas que realizan los cabezales durante su paso por la superficie de los platos magnéticos. Este servomecanismo se basa en señales de control y balizas repartidas a lo largo y ancho de todas las superficies del disco duro.
7. Un convertidor analógico-digital capaz de transformar la señal amplificada de los cabezales en su versión digital, y decodificarla en su caso.
8. Una interfaz para intercambiar información con un *host* o sistema como un computador personal. En los discos duros de escritorio comunes se trata de una interfaz ATA en serie o paralelo.
9. Un programa interno o firmware que gobierne el estado y el comportamiento de cada uno de los componentes del disco, que implemente el protocolo de comunicaciones definido en el estándar ATA/ATAPI vigente, y que dote de funcionalidad al dispositivo. Es esta pieza de software de los discos duros la gran desconocida, y en la que nos centraremos a continuación.

3.2. Firmware de un disco duro actual

Todo disco duro tiene un programa interno de funcionamiento al que conocemos como firmware [14] [15], que se encarga tanto de mediar en las comunicaciones entre el dispositivo y el ordenador al que se conecta (identificación, protocolo de

comunicación, etc.), como de administrarlo internamente (rutinas de arranque, corrección de problemas, respuesta antes determinados eventos, etc.). El programa, consistente en múltiples piezas de código cada una de las cuales se encarga de una tarea específica, viene instalado de fábrica y es imprescindible para el funcionamiento del disco duro. No se trata de un programa accesible de una forma normal, ni un programa residente en la zona de datos normalmente accesible, sino de un programa que funciona de forma totalmente transparente al usuario, a la vez que inaccesible.

El firmware de un disco duro actual es un programa complejo que efectúa un gran número de actividades: en primer lugar, tiene una implementación completamente funcional del protocolo ATA/ATAPI que le permite comunicarse con cualquier dispositivo adherido al estándar que lo define. El protocolo dicta un número de obligaciones que deben estar registradas en la implementación del protocolo, haciendo de ésta una parte pesada del mismo (en términos de cantidad de información que es necesario almacenar como memoria de programa). Además del protocolo de comunicaciones al completo, el firmware debe resolver de alguna forma las siguientes tareas:

a. El control del posicionamiento y estabilidad de los cabezales sobre la superficie del medio magnético
b. El control de la altura a la que flotan los cabezales sobre cada superficie
c. Las rutinas de lectura y escritura, y de verificación, incluyendo la traducción de coordenadas lógicas (CHS) en coordenadas físicas en la superficie de los platos magnéticos
d. El control de la velocidad de giro del motor mediante un servomecanismo
e. El tratamiento digital de la señal de lectura mediante diferentes modelos estadísticos
f. Detección y corrección de errores

g. Gestión de defectos irrecuperables

h. Rutinas de emergencia (apagado, sensor de caída libre)

Toda esta cantidad de información hace que la parte que denominaremos la memoria de programa estática sea por sí sola y relativamente hablando *bastante grande*: desde ~0,5 MB hasta varios MB, dependiendo del fabricante[3]. Los modelos de acceso al medio magnético en lo que se refiere a la zona de datos de usuario forman parte de lo que se conoce como el *hyper-tuning* de los discos duros, que tiene registro en forma de *parámetros adaptativos* únicos para cada disco duro del mercado, y que son resultado de exhaustivas pruebas que se hacen en fábrica a cada unidad de disco duro en la cadena de producción [16].

Por otro lado existe lo que denominamos la parte *dinámica* del firmware: es un conjunto de información que varía con el uso del disco duro, consistente principalmente en listas de sectores o pistas y listas o programas de traducción: listas de sectores que cuya lectura ha resultado defectuosa (sectores marcados como candidatos a defectuosos); listas de sectores defectuosos [17] [18], tablas de relación entre sectores defectuosos y sectores de repuesto o reubicados; tablas de sectores marcados como defectuosos en el formateo a bajo nivel realizado en fábrica; tablas de traducción entre direcciones CHS y LBA y programas que ejecutan esta traducción…

El peso relativamente grande del firmware del disco duro, así como la naturaleza variable de parte del mismo, ha supuesto un reto en el diseño de los mismos que los diferentes fabricantes han optado unánimemente por solucionar reservando una parte de la zona usable de los platos magnéticos del mismo para almacenar esta información; parte que denominamos *área reservada del disco duro*. La forma en que cada fabricante aprovecha esta zona reservada es diferente y queda totalmente fuera del alcance del protocolo ATA/ATAPI, por lo que no existe consenso sobre cómo debe

[3] Sirva como referencia el tamaño de las copias de seguridad de las zonas reservadas que se pueden generar con ayuda de las herramientas mencionadas en el Capítulo 2.

organizarse la información en dicho área, ni existe información pública sobre cómo acceder a esta zona ya que queda en manos del fabricante [19]. La información acerca de esta zona del disco duro que se detalla en este documento ha sido obtenida directamente de los manuales y los recursos de los fabricantes de las herramientas enumeradas en el Capítulo 2, y mediante análisis de soluciones y técnicas utilizadas actualmente [47].

3.2.1. Organización interna del firmware del disco duro

Todos los fabricantes actuales de discos duros dividen el firmware del disco en dos partes fundamentales [20]:

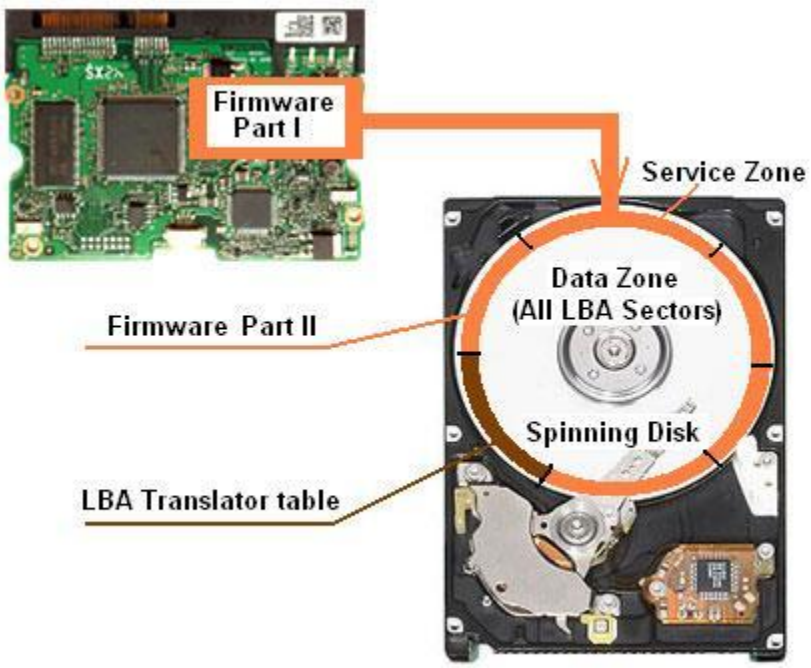

Figura 3: Databe. *How Hard Drive Works: Firmware on Disk Platter and PCB*. Extraído de <http://www.databe.com/articles/article6.html>

A. Firmware estático en memoria ROM/EEPROM/Flash (Figura 3, *Part I*): Consiste en la parte básica del firmware, con las instrucciones básicas para inicializarlo, y la información necesaria para poder realizar accesos al área de sistema,

incluyendo la ubicación de la misma dentro de las superficies de los platos magnéticos.

B. Firmware estático en superficies y firmware dinámico, también en superficies (Figura 3, *Part II*). Algunos fabricantes optan por almacenar parte del código de funcionamiento sobre los platos magnéticos en lo que se conoce como los *overlays* de código. Los fabricantes Western Digital o Seagate, citando algunos ejemplos, almacenan la implementación del protocolo ATA en las superficies, así como otras partes del código de funcionamiento.

3.2.2. Firmware estático en memoria ROM/EEPROM/Flash

Figura 4: Silicon Storage Technology. *SILICON STORAGE TECHNOLOGY SST25VF040B-50-4I-S2AF 4M FLASH MEMORY, SPI EEPROM, SOIC8*. Extraído de <http://cpc.farnell.com/silicon-storage-technology/sst25vf040b-50-4i-s2af/4m-flash-memory-spi-eeprom-soic8/dp/SC09630>

Los discos duros presentes en el mercado tienen un programa de arranque instalado bien en la propia memoria ROM disponible en el circuito integrado del microcontrolador principal (*ROM interna*), o bien en un en un integrado externo al microcontrolador, instalado en la placa electrónica en forma de memoria ROM (EEPROM, NVRAM, Flash, etc.) y de capacidad limitada —habitualmente de entre 256 kB y 512 kB (*ROM externa*)—, similar al mostrado en la Figura 4. Aunque es posible reprogramar esta memoria usando comandos ATA específicos del fabricante en gran

parte de los discos duros presentes en el mercado[4], el espacio disponible es muy limitado (unos pocos kilobytes), ya que está prácticamente todo el espacio ocupado por parte del firmware estático, y el controlador que se propone desarrollar en este documento para obtener acceso a las áreas restringidas del disco no toma esta memoria en consideración.

La memoria ROM contiene a su vez las direcciones físicas, en los platos magnéticos, donde reside el resto del firmware, que el disco duro deberá leer durante la secuencia de arranque para permitir acceso a la información de usuario. Así mismo contiene información suficiente para poder comenzar a operar los cabezales de lectura y escritura, y hacerles salir de su zona de aparcamiento para buscar las pistas de arranque (las áreas de sistema o restringidas). En algunos fabricantes esta memoria contiene el modelo exacto de dispositivo y su número de serie, aunque la tendencia es almacenar este último en las superficies magnéticas [21]. Otra información relevante e imprescindible conocer por el firmware del disco duro durante la secuencia de arranque es almacenada en la memoria ROM, como el número de cabezales de lectura/escritura activos, versión del código (para compatibilidad con la parte del código en platos magnéticos), o el tipo de preamplificador con el que debe comunicarse [22].

[4] Prueba de ello es que algunos fabricantes proporcionen actualizaciones de firmware, que reprograman la memoria EEPROM o Flash del dispositivo: http://knowledge.seagate.com/articles/en_US/FAQ/004559en?language=en_US

3.2.3. Firmware estático y dinámico en superficies

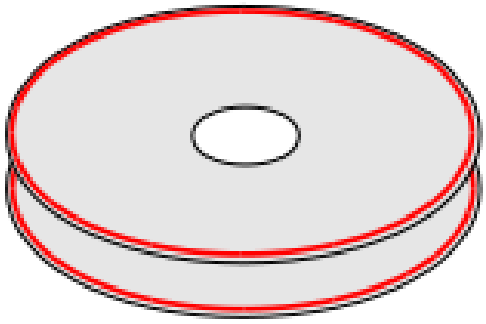

Figura 5: *Áreas reservadas en las superficies de los platos magnéticos*. Elaboración propia.

La mayor parte del firmware del disco reside en los platos magnéticos. La ubicación exacta en los platos magnéticos depende del fabricante; en la mayoría de discos duros, incluidos los del fabricante Western Digital, es habitual que se encuentre en una corona exterior [23] (ver Figura 5). De forma general, podemos decir que un disco duro actual tiene 2 particiones físicas (con propiedades geométricas que pueden ser distintas), que separan la **zona de datos de usuario** de la **zona de datos de sistema**[5] (en Figura 6, cilindros 0 y positivos, y cilindros negativos, respectivamente).

[5] Referencias a las particiones de usuario y de sistema se pueden encontrar en los manuales de diagnóstico de los discos duros Seagate F3, comando 'm' de nivel T (*format partition*). Estos manuales no están disponibles para su compra y solo se pueden obtener en Internet; a fecha de 28 de septiembre de 2015 aún se puede consultar en: http://www.scribd.com/doc/230400927/F3-SerialPort-Diagnostics

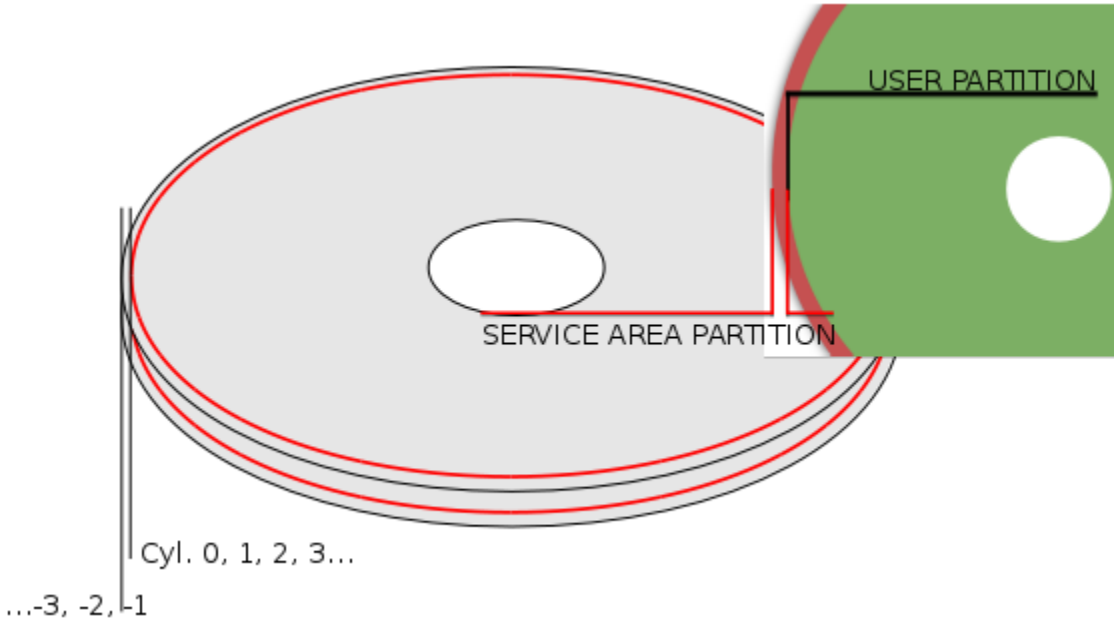

Figura 6: *Particiones físicas en la superficie de un disco magnético*. Elaboración propia.

Cabe recalcar que estas particiones no tienen nada que ver con las particiones que puede hacer el usuario en un disco estándar, que particionan lógicamente el área de datos de usuario y separan sistemas de ficheros distintos [24] [25]. En algunos fabricantes, la partición de sistema está en lo que se denominan *cilindros negativos*; es decir, los sectores de la partición de sistema tienen coordenadas CHS físicas del tipo (c, h, s), siendo c un entero menor que 0 (ver Fórmula 1):

$$sector = (c, h, s);$$
$$\{h \in \mathbb{N} \mid 0 \le h \le maxH\}, \{s \in \mathbb{N} \mid 1 \le s \le SPT\}, \{c \in \mathbb{Z} \mid minC \le c \le -1\} \quad \text{[Fórmula 1]}$$

Siendo *maxH* el número de cabezales de lectura activos, *SPT* el número de sectores por pista, y *minC* el cilindro negativo más alejado del eje de giro de los platos.

Esta parte del firmware del disco duro está organizada en diferentes módulos [26] [27], que contienen la siguiente información:

a) Un directorio de módulos. Es el índice que señala dónde están almacenados cada uno de los otros módulos; su implementación varía según cada fabricante.
b) Overlays de código. Parte del programa de funcionamiento del disco duro (como la implementación del protocolo ATA) reside en estos *overlays*.
c) Lista de defectos primarios (P-List, T-List). Durante el primer formateo en fábrica se detectan irregularidades en las superficies magnéticas y se descartan sectores e incluso pistas defectuosas. Según familia de discos duros, los sectores defectuosos se almacenan en una lista única o en una combinación de listas, para posterior uso (ver punto *d)* a continuación).
d) Lista de defectos *incrementales* o secundarios (G-List). Durante el tiempo de vida útil del disco se pueden registrar diversas condiciones de error al tratar de acceder a determinados sectores, pudiendo el firmware del disco darlos por inservibles y descartarlos, sustituyéndolos por otros de reserva.
e) Lista de sectores candidatos a ser defectivos. Los sectores que han dado algún problema pero que se entiende que aún contienen información útil (es decir, han dado fallo en lectura y aún no se ha tratado de sobrescribir) se almacenan en esta lista, preparados para ser dados de baja definitivamente cuando se trate de escribir en ellos de nuevo.
f) Tabla y código de traducción CHS ⇔ LBA. Esta traducción no es una implementación tal como la definida en el estándar ATA/ATAPI-5 (obsoleta en ATA/ATAPI 7 [28]), sino un mecanismo interno del disco duro para tener en cuenta, en la traducción de sectores físicos a sectores lógicos, los defectos registrados en las listas de defectos [29].
g) Parámetros de configuración: Administración Acústica Automática (*AAM*, de sus siglas en inglés); Administración Automática de Energía (*APM*, de sus siglas en inglés); áreas protegidas del huésped o *HPA*; *overlays* de configuración de dispositivo o *DCO*…

h) Scripts de self-test. No todos los fabricantes dejan esta información en las unidades que salen para venta al público, ya que pueden revelar información a sus competidores sobre todas las pruebas que pasa un disco antes de estar disponible, y sobre los mecanismos de *tuning* y calibrado.
i) Parámetros adaptativos de lectura y escritura. Forman parte de los modelos estadísticos que se usan en la detección/análisis de la señal de lectura y la generación de la señal de escritura, únicos para cada disco duro del mercado. Cada disco duro se calibra individualmente, obteniendo resultados diferentes que se codifican en estos módulos.
j) Información de seguridad. El estándar ATA/ATAPI define la posibilidad [30] de añadir una capa de seguridad (bloqueo) de los dispositivos, mediante una contraseña que se almacena en sus módulos respectivos en la superficie de los platos magnéticos.
k) Información SMART (Self-Monitoring, Analysis, and Reporting Technology). El estándar ATA/ATAPI define un set de características adicionales diseñado para que el dispositivo sea capaz de alertar en caso de fallo inminente (alta degradación, detección de diferentes condiciones de error, etc.). Si un dispositivo implementa este set de características, debe almacenar registros del estado del dispositivo que sirvan a este propósito; los discos duros actuales almacenan estos registros en los platos magnéticos en el módulo SMART (o equivalente).

3.3. El caso del fabricante Western Digital

Para el desarrollo del controlador propuesto en el presente proyecto, de todos los fabricantes de discos duros nos vamos a centrar en uno: Western Digital. La compañía (Western Digital Company o *WDC*) acaparaba en 2010 alrededor de un 30% de la cuota de mercado en discos duros tradicionales (mecánicos) [31] [32]. Dado que cada fabricante utiliza sus propios mecanismos internos, y su propia forma de organizar la

información reservada en el interior de sus discos duros, un driver completo requeriría una investigación del modo de funcionamiento de cada uno de los modelos de discos en el mercado; en este documento tomaremos como referencia este fabricante y el driver se desarrollará para interactuar con discos Western Digital, concretamente de la superfamilia Marvell (equipados con microcontrolador Marvell; son los discos duros comerciales fabricados por Western Digital desde 2001 y hasta la fecha en que se escribe el documento [33]).

El firmware en discos duros Western Digital modernos se organiza, en líneas generales, en dos segmentos: el segmento EEPROM y el segmento en superficies [34], que se detallan a continuación.

3.3.1. Segmento EEPROM

La información contenida de forma estática en lo que se conoce como ROM (memoria de solo lectura) puede estar almacenada internamente en el espacio ROM del microprocesador Marvell del que disponen estos discos duros, o externamente en una memoria Flash dedicada. La capacidad de este segmento es variable, habiendo observado tamaños desde 128 kB hasta 256 kB en el caso de memorias ROM internas, y desde 256 kB hasta 512 kB en memorias Flash externas.

Este segmento del firmware contiene la secuencia de arranque del disco duro y los siguientes módulos:

a. Directorio de módulos, con las direcciones dentro de la memoria ROM del resto de módulos. La mayoría de modelos almacena una copia de respaldo de este directorio.
b. Mapa de cabezas o *head map*. Contiene una lista de los cabezales del disco duro activos. Durante el ensamblaje y las pruebas de rendimiento y calidad puede que algunas superficies del disco duro no hayan resultado lo bastante buenas como para ser formateadas. En estos casos, se desactivan dichas

superficies, quedando el cabezal correspondiente desactivado. Esta operación se realiza también en el reacondicionamiento de discos duros para ser puestos de nuevo a la venta [35].

c. Ubicación del área de sistema en la superficie en coordenadas ABA, para cada una de las superficies, y extensión de la misma.
d. Listado de defectos en el área de sistema y tabla de traducción de direcciones, que tenga en cuenta estos defectos. El área de sistema suele tener muy pocos defectos ya que se tiene especial cuidado con la elección de la zona en que se ubica al ser una zona crítica.
e. Parámetros de lectura y escritura en el área de sistema. Las operaciones de lectura y escritura en esta área son más sencillas que en el área de usuario dado que tienen una densidad de datos menor[6]; los parámetros de lectura y escritura para el área de datos de usuario suman una cantidad de información mayor y están ubicados en el área de sistema.

3.3.2. Segmento en superficies

Una vez el disco duro ha comenzado la secuencia de arranque y sabe dónde ir a buscar el resto de información que le falta para quedar operativo, la unidad saca los cabezales de la zona de aparcamiento y trata de localizar el área de sistema. Las unidades Western Digital almacenan por defecto 2 copias del área de sistema [36] [37]: si el disco duro tiene al menos 2 superficies activas, se almacena cada copia en una superficie (superficies 0 y 1). Si es una unidad con solo una superficie activa, ambas copias se almacenan en la misma superficie, en distintas ubicaciones.

Por defecto, la superficie más inferior es la que tiene la copia del área de sistema denominada *copia 0*. Se muestra en la Figura 7 un diagrama de la numeración de las

[6] Esta información se puede obtener de las tablas de zonas de un disco duro, usando alguna de las herramientas mencionadas en el Capítulo 2; se observa que hay un SPT significativamente menor en el área de sistema que en el área de usuario.

superficies magnéticas en un disco duro, junto con sus respectivos cabezales de lectura/escritura:

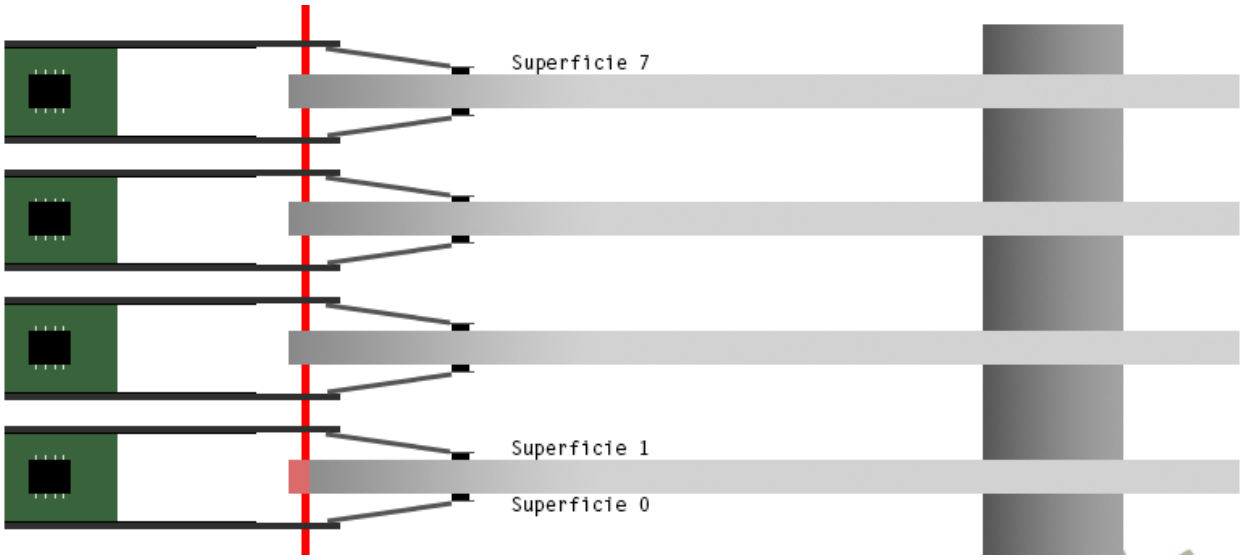

Figura 7: *Cabezales y superficies en un disco duro junto a su numeración*. Elaboración propia.

En su estado normal y sin ayuda externa, el disco duro no es capaz de arrancar si no tiene suficientemente sana la copia 0. Una vez localizado el sector absoluto ABA en donde se encuentran estas estructuras, comienza la lectura de la información necesaria para el arranque:

a. El directorio de módulos en superficie (ver Figura 8). Contiene direcciones del resto de módulos. Suele ser el módulo con ID = 1.

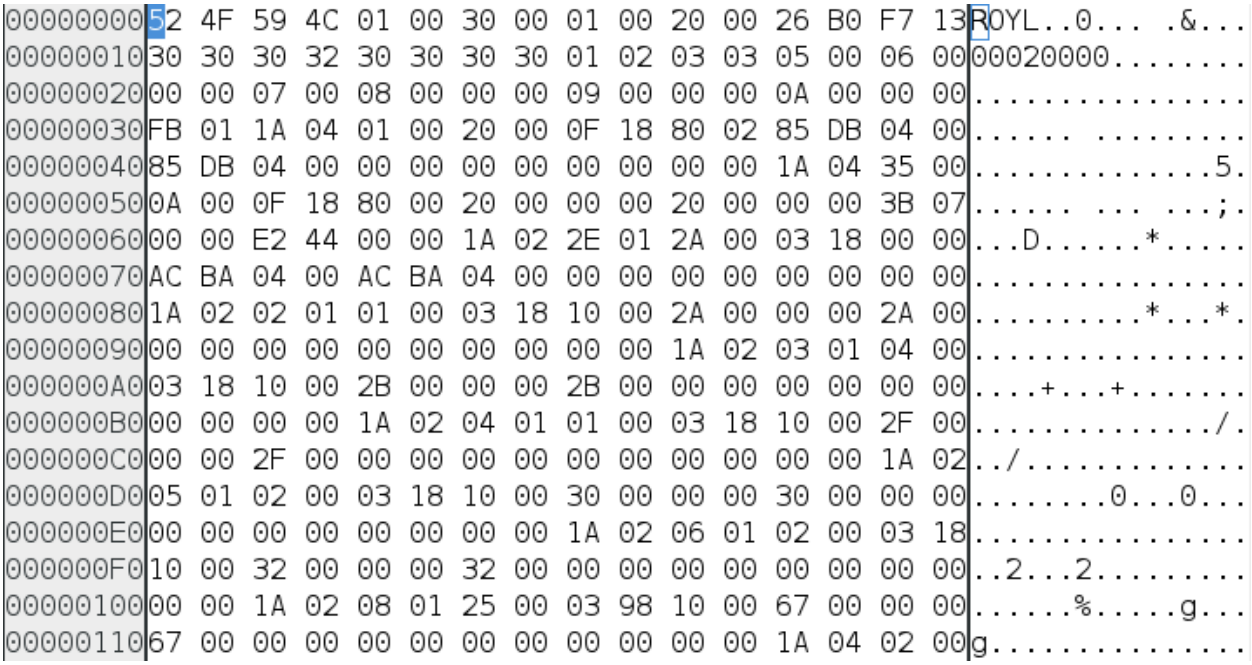

Figura 8: *Módulo 1: directorio de módulos*. Elaboración propia.

b. La identificación del dispositivo. Contiene el modelo y el número de serie con el que se debe identificar (ver Figura 9). Suele ser el módulo con ID = 2.

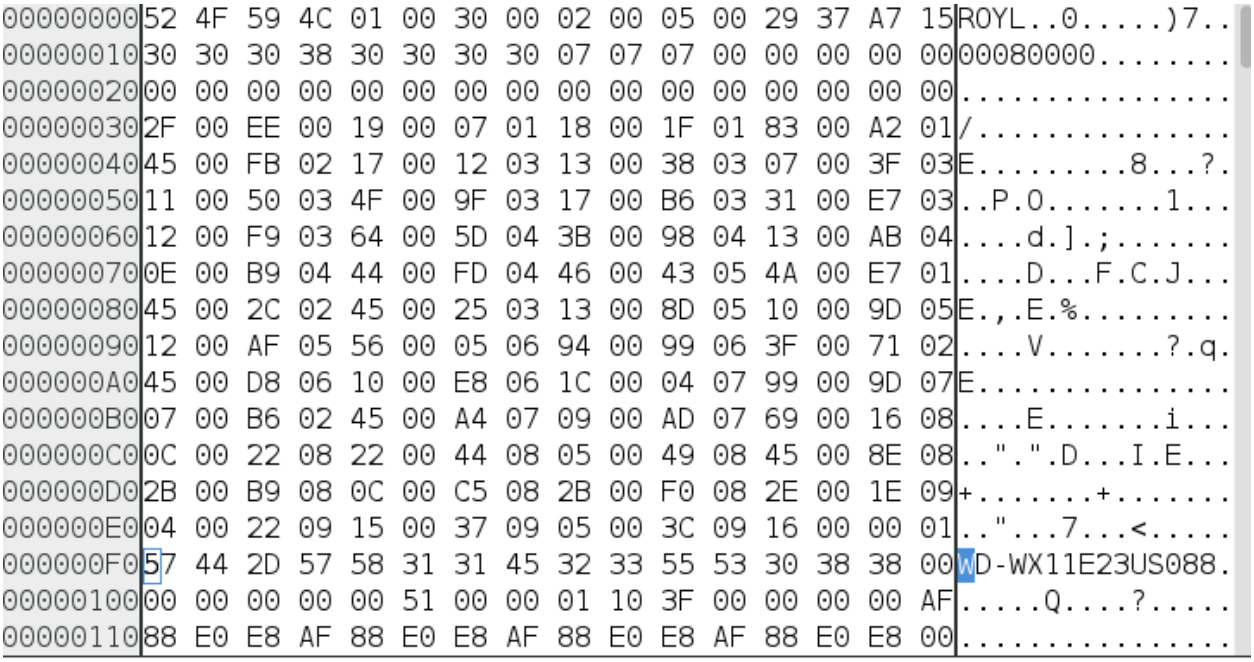

Figura 9: *Módulo 2: identificación del dispositivo*. Elaboración propia.

c. La información de seguridad: estado de bloqueo, contraseñas de acceso y claves de cifrado si está disponible, etc.
d. Parámetros de lectura y escritura en área de usuario.
e. Los overlays de código (overlay ATA y resto del código de funcionamiento); suelen ocupar varios módulos.
f. El traductor de direcciones o *Translator*. Imprescindible para ganar acceso al área de datos de usuario.
g. La tabla de sectores pendientes de reubicar. Durante el arranque, los sectores marcados como defectuosos y pendientes de reubicar se tratan de reubicar a direcciones sanas (sectores de repuesto).
h. La tabla SMART.
i. Parámetros de funcionamiento configurables, como AAM y APM.

Además de la información necesaria para el arranque, el disco duro almacena en forma de módulos:

a. Registros o *logs* de fábrica, de calibrado y de otros eventos;
b. *Self-test scripts* o rutinas de auto testeo y calibrado (usados en fábrica para calcular todos los parámetros de funcionamiento);
c. Las tablas de defectos usadas para calcular el *Translator*;
d. Información del fabricante, tal como la denominación de la familia a la que pertenece el disco duro, un listado de modelos incluidos en la misma familia, el fabricante de los platos magnéticos, el fabricante de los cabezales de lectura y escritura…, y
e. Módulo de pruebas: cada cabezal dispone de un módulo para hacer pruebas de integridad de escritura y lectura, sin comprometer ninguna otra zona en la escritura.

El segmento del firmware almacenado en superficie es el que nos es de especial interés para el desarrollo de nuestro controlador de disco, y es el que nos lleva a la siguiente sección.

3.3.3. Espacio mínimo en desuso del área de servicio

Por lo anterior, sabemos que cada disco duro Western Digital moderno tiene 2 copias del área de sistema, ubicadas habitualmente (sin haber encontrado excepciones) en las superficies 0 y 1 del primer plato, o plato inferior. Hay dos detalles cruciales en este aspecto para el diseño de nuestro driver:

a. Los discos duros escriben secuencialmente en cada una de las superficies disponibles, y mueven todos sus cabezales a la vez, en bloque [38]. El paso de una superficie a la siguiente se da tras la escritura continua de un determinado número de sectores en una única superficie; la cantidad de sectores que se escriben de seguido en una superficie es un parámetro a obtener del área de servicio del disco duro, y por los firmwares analizados, en discos duros Western Digital modernos tiene un valor medio de 150000 sectores (~300 MB). Cuando se da el cambio de pista, el disco duro está diseñado para continuar la lectura en la nueva pista perdiendo el mínimo tiempo posible; es decir:
   - ➢ El cabezal ha de desplazarse hasta la siguiente pista en el menor tiempo posible. Esto se consigue con diferentes técnicas y el uso de pistas de servo, que sirven de balizas para el posicionamiento del cabezal en cada pista.
   - ➢ Una vez posicionado el cabezal sobre la nueva pista, el tiempo de espera hasta dar con el siguiente sector a leer debe ser mínimo. Es fácil de comprobar que si las pistas estuvieran completamente alineadas unas con otras, al desplazarse el cabezal de una pista a la siguiente, tendría que esperar casi una revolución del plato completa hasta encontrarse con el siguiente sector a leer. Por este motivo, existe un ligero desplazamiento del comienzo del sector relativo a la pista anterior, para mantener una velocidad de lectura o escritura estables.

- ➢ La distancia desde el eje de giro sobre la que está el cabezal antes de un cambio de superficie debe variar lo mínimo posible. Si en el cambio de superficie los cabezales de lectura tuvieran que alejarse significativamente del lugar en el que están (es decir, que el brazo actuador tuviera que desplazar mucho el cabezal), se perdería mucho tiempo buscando y sincronizando con la nueva pista, alejada en la geometría del disco duro.

b. Los discos duros Western Digital con daños en el área de servicio (sectores dañados) cuentan con medios para ser reparados moviendo los módulos dañados (o copias funcionales) a alguna de las superficies superiores (2, 3, 4…) [39].

De esto podemos extraer que, de forma general, un disco duro Western Digital con más de dos superficies activas tiene libres las pistas de las superficies 2 y superiores que están en los mismos cilindros que las superficies 0 y 1 tienen reservadas para el área de servicio [Fig. 10].

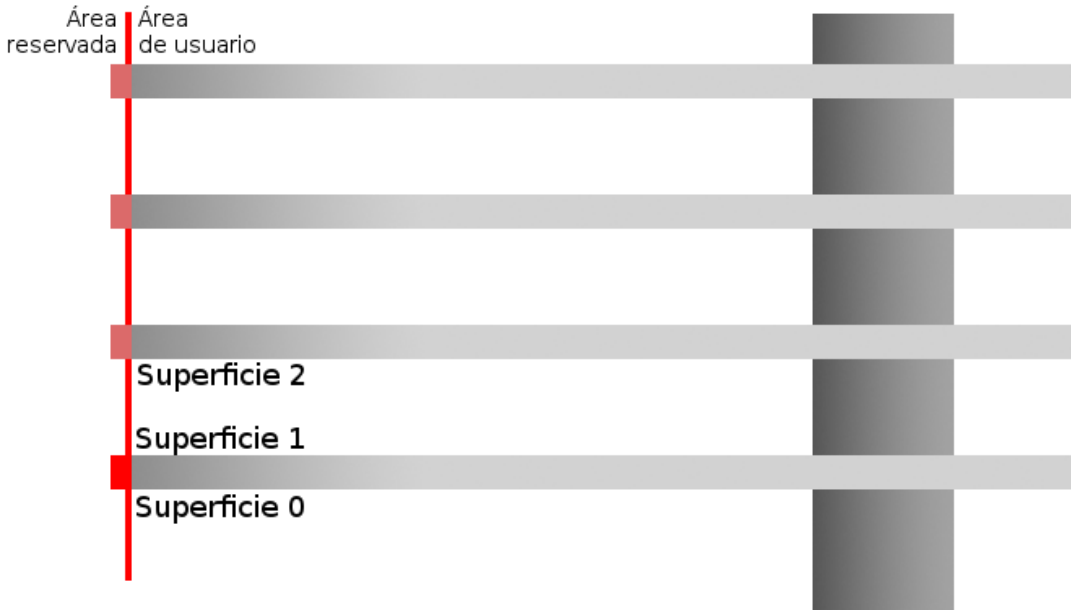

Figura 10: *Verticalidad e igual distribución de las zonas restringidas*. Elaboración propia.

En las hojas de especificaciones de los discos duros de Western Digital podemos encontrar el número de platos y/o superficies activas en cada uno [40]; con esta información podemos empezar a hacernos una idea de cuánto espacio disponible tenemos actualmente en desuso. Sin conocer más detalles sobre el espacio no usado en el área de servicio en una *superficie de sistema* (superficies 0 y 1), por ahora podemos decir que tenemos disponible como mínimo N veces la suma de los tamaños de todos los módulos contenidos en la superficie 0, siendo N el número de cabezales activos del disco menos 2 (ver Fórmula 2):

$$C_{min} = (H_{max} - 2) \times S_0 \qquad \text{[Fórmula 2]}$$

Siendo C_{min} la capacidad mínima disponible, H_{max} el número de cabezales de lectura activos, y S_0 el tamaño del área de servicio en la superficie 0.

A modo de ejemplo, si tenemos un disco duro con 6 cabezales en uso, con un tamaño de área de sistema de 100 MB, tendremos disponibles 4 * 100 MB = 400 MB de espacio utilizable.

3.3.4. Acceso al área de servicio en discos duros Western Digital Marvell

En este apartado veremos una técnica disponible para comunicarse con el área de sistema en discos duros Western Digital, que hace uso de los llamados *Vendor Specific Commands* o VSC. La rutina de acceso al área de servicio es como sigue:

a. Enviar al disco duro el comando conocido como *VSC Enable* o *Super On*: es una llave especial que hace que el disco duro habilite el procesado de comandos que tratan con el área de sistema u otra información privilegiada. En las familias de Western Digital Marvell, este comando se puede enviar al disco del siguiente modo [41] (se utiliza la llamada de sistema en Linux ***outb*** [42], que escribe el byte indicado —primer parámetro— en el puerto indicado —segundo

parámetro—. Se omite el valor de p1-p7 ya que hay que definirlo para cada caso:

```
outb (0x45, p1);
outb (0x00, p2);
outb (0x00, p3);
outb (0x44, p4);
outb (0x57, p5);
outb (0xA0, p6);
outb (0x80, p7);
```

b. Enviar al disco duro el VSC que se requiera: leer sectores, escribir sectores, leer páginas de configuración, etc. (ver a continuación).
c. Una vez enviado el VSC, si el comando enviado es de lectura, prepararse para recibir los datos; si es de escritura, enviarlos (ver a continuación).
d. Enviar al disco duro otra llave, *VSC Disable* o *Super Off*, para indicar que hemos terminado con los comandos de fabricante y podemos continuar con la operación normal. El comando es muy similar al primero, *VSC Enable*; solo difieren en un bit [43]:

```
outb (0x44, p1);
outb (0x00, p2);
outb (0x00, p3);
outb (0x44, p4);
outb (0x57, p5);
outb (0xA0, p6);
outb (0x80, p7);
```

Una vez el disco está en modo *VSC Enable*, nos podemos comunicar con él con un juego de comandos específicos, diferentes de los definidos en el estándar ATA/ATAPI pero no incompatibles con él: el estándar deja *hueco* para comandos reservados, que usan los fabricantes para estas operaciones. La forma en que nos comunicamos es encapsulando un comando específico del fabricante en un comando ATA estándar, en concreto el comando SMART WRITE LOG, con la siguiente estructura [44]:

```
outb(0xd6, p1);          //Escribir log SMART (ATA std.)
outb(0x01, p2);          //Número de sectores = 1
outb(0xbe, p3);          //dirección del log SMART
outb(0x4f, p4);          //Siempre 0x4F (ATA std.)
outb(0xc2, p5);          //Siempre 0xC2 (ATA std.)
outb(0xa0, p6);          //Puerto IDE maestro
outb(0xb0, p7);          //Comando SMART (ATA std.)
…
write_sector(...);
```

El tercer byte que escribimos en el puerto es el reservado para la dirección del log de la tabla SMART. Según la siguiente tabla (Tabla 1) extraída del estándar ATA/ATAPI-7 [45]:

Log address	Content	R/W
00h	Log directory	RO
01h	Summary SMART error log	RO
02h	Comprehensive SMART error log	RO
03h	Extended Comprehensive SMART error log	See note
04h-05h	Reserved	Reserved
06h	SMART self-test log	RO
07h	Extended self-test log	See note
08h	Reserved	Reserved
09h	Selective self-test log	R/W
0Ah-1Fh	Reserved	Reserved
20h	Streaming performance log	See note
21h	Write stream error log	See note
22h	Read stream error log	See note
23h	Delayed sector log	See note
24h-7Fh	Reserved	Reserved
80h-9Fh	Host vendor specific	R/W
A0h-BFh	Device vendor specific	VS
C0h-FFh	Reserved	Reserved

Key –
RO - Log is read only by the host.
R/W - Log is read or written by the host.
VS - Log is vendor specific thus read/write abiltiy is vendor specific.

NOTE - Log addresses 03h, 07h, 20h, 21h , 22h, and 23h are used by the READ LOG EXT and WRITE LOG EXT commands. If these log addresses are used with the SMART READ LOG command, the device shall return command aborted.

Tabla 1: Dirección de SMART log estandarizadas. Extraído del estándar ANSI INCITS 397-2005 ATA/ATAPI-7 (Table 20: *Log address definition*. p. 190.)

El estándar reserva las direcciones de log SMART `0xa0` a `0xbf` para uso específico del fabricante (direcciones marcadas con *VS* o *Vendor Specific*). En este caso, el valor que se utiliza en este byte para completar el comando SMART WRITE LOG es el valor **`0xbe`**, que marca este comando como puente para enviar un segundo comando específico de fabricante [46]: tras enviar este comando, el disco duro espera recibir una página de log de tamaño 1 sector, con el comando completo: flag de lectura/escritura, coordenadas, etc.

3.3.4.1. Lectura

Para leer información del área de servicio, usaremos una página de log como la siguiente:

```
word 0: comando (VSC). Para acceso a cilindros negativos
usamos 0x000c [47]
word 1: 0x0001 para lectura
word 2: coordenadas CHS: C (LSW)
word 3: coordenadas CHS: C (MSW)
word 4: coordenadas CHS: H
word 5: coordenadas CHS: S
word 6: longitud (cantidad de sectores)
words 7-256: padding (0x00)
```

Todas las palabras de 2 bytes han de escribirse en *Little Endian* (LE) [48], por lo que la cadena escrita para leer el sector en coordenadas CHS (-1, 2, 33) queda como sigue:

```
0x0c 0x00 0x01 0x00 0x01 0x00 0x00 0x00
0x02 0x00 0x32 0x00 0x01 0x00 0x00 0x00
0x00 … padding
```

Nótese que el cilindro en el cual se quiere escribir queda positivo; los comandos que afectan al área de servicio se sobreentiende que afecta a los cilindros negativos (no

se puede leer usando estos comandos el área de datos de usuario; es decir, los cilindros 0 y positivos).

Una vez enviado el comando VSC, antes de poder leer los datos requeridos hay que enviar otro comando estándar, pero con un parámetro *vendor specific* —en este caso con valor **0xbf**—: el comando SMART READ LOG, que se muestra a continuación:

```
outb(0xd5, p1);          //Leer log SMART (ATA std.)
outb(scount, p2);        //Número de sectores a leer
outb(0xbf, p3);          //dirección del log SMART
outb(0x4f, p4);          //Siempre 0x4F (ATA std.)
outb(0xc2, p5);          //Siempre 0xC2 (ATA std.)
outb(0xa0, p6);          //Puerto IDE maestro
outb(0xb0, p7);          //Comando SMART (ATA std.)
```

Tras lanzar el comando SMART READ LOG, podemos quedarnos a la escucha, esperando a que el disco duro devuelva la información requerida, y leyéndola mediante la llamada de sistema ***insw*** [49], proporcionando como parámetros el puerto IDE correspondiente, un buffer de lectura y un tamaño de 256 palabras (512 bytes), si se ha indicando en el VSC que se quería leer 1 sector.

3.3.4.2. Escritura

Para escribir información en el área de servicio, usaremos una página de log como la siguiente:

```
word 0: comando (VSC). Como en el caso de lectura, usamos
0x000c.
word 1: 0x0002 para escritura
word 2: coordenadas CHS: C (LSW)
word 3: coordenadas CHS: C (MSW)
word 4: coordenadas CHS: H
word 5: coordenadas CHS: S
word 6: longitud (cantidad de sectores)
```

```
words 7-256: padding (0x00)
```

Al igual que en el caso de lectura, todas las palabras de 2 bytes han de escribirse en LE, quedando la escritura del sector cuyas coordenadas CHS son (-1, 2, 33) como sigue:

```
0x0c 0x00 0x02 0x00 0x01 0x00 0x00 0x00
0x02 0x00 0x32 0x00 0x01 0x00 0x00 0x00
0x00 … padding
```

Al igual que en el caso anterior, el cilindro se sobreentiende que es de la zona negativa, y se escribe sin signo. Del mismo modo, una vez enviado el VSC mediante la página anterior (512 bytes), antes de enviar los datos, es necesario avisar al disco duro de que estamos preparados para transferir datos, usando el mismo comando SMART READ LOG que en el caso de lectura:

```
outb(0xd5, p1);          //Leer log SMART (ATA std.)
outb(scount, p2);        //Número de sectores a leer
outb(0xbf, p3);          //dirección del log SMART
outb(0x4f, p4);          //Siempre 0x4F (ATA std.)
outb(0xc2, p5);          //Siempre 0xC2 (ATA std.)
outb(0xa0, p6);          //Puerto IDE maestro
outb(0xb0, p7);          //Comando SMART (ATA std.)
```

Tras lanzar este comando SMART READ LOG, el disco duro espera recibir por el puerto de entrada/salida el sector o los sectores anunciado(s). Para escribir el sector, se utiliza la llamada de sistema ***outsw*** [50], proporcionando como parámetros el puerto de salida, un puntero a la estructura de datos que contiene nuestro sector (un vector de 256 palabras o 512 bytes), y la longitud de la cadena de datos a escribir.

3.3.4.3. Geometría del disco duro y del área de servicio

Un factor crucial a la hora de diseñar un controlador para trabajar con el espacio sobrante del área de servicio de un disco duro es su geometría. En particular, nos interesa:

a. Las superficies activas disponibles, así como su numeración (el mapa de cabezales).
b. Los cilindros negativos reservados para el área de servicio.
c. La cantidad de sectores por pista (el parámetro STP) en la zona del área de servicio.

Ninguno de los tres datos necesarios es información que se pueda encontrar en la hoja de especificaciones del fabricante, ni es información que se pueda derivar de ella. Esto es así porque:

a. La existencia de cabezales de lectura instalados en el dispositivo no garantiza que estén activados (pueden haber sido desactivados por diferentes motivos).
b. La numeración física de los cabezales disponibles no tiene por qué corresponderse con la numeración lógica de los mismos (el parámetro usado para referirse a ellos).
c. Los cilindros reservados en el área de servicio no forman parte de los contemplados en las hojas de especificaciones, donde solo se hace referencia a los cilindros pertenecientes a la partición de usuario[7].
d. El parámetro STP no es único para un disco duro; depende del cabezal y la ubicación o distancia radial en los platos magnéticos [51].

[7] Esta información se puede extraer de las hojas de especificaciones de discos duros de poca capacidad (menores de 8 GB), que especifican su capacidad tanto en LBA como en CHS. La capacidad garantizada al usuario en sectores es la cantidad de LBA anunciados; la cantidad de LBA también se puede extraer de la información de geometría que anuncian estos discos con la siguiente fórmula:

$$n = c \times h \times spt$$

Mediante esta fórmula se puede extraer que la capacidad anunciada al consumidor en LBA coincide con los cilindros usables anunciados.

Esta información se puede obtener del disco duro mediante otros comandos VSC, que son los de obtención de **páginas de configuración**. Conociendo estos parámetros, podremos saber el tamaño exacto del área de servicio, y podremos construir un algoritmo que proporcione un acceso unidimensional a los sectores, requisito imprescindible para comunicarnos con el kernel de Linux y proporcionar al sistema acceso al mismo, como si de cualquier otro dispositivo de almacenamiento se tratase [52].

Para extraer esta información del disco duro, durante la investigación realizada con diferentes modelos de discos duros, se han encontrado tres arquitecturas de disco diferentes en discos duros Western Digital de los últimos 10 años, que son las siguientes (en incidencia, por orden cronológico):

- Western Digital Marvell
- Western Digital Marvell ROYL
- Western Digital Marvell ROYL 20B

Los nombres que reciben estas arquitecturas tienen carácter meramente identificativo y pueden no corresponderse con los nombres con los que los identifica el fabricante. La justificación de los nombres está en que a ciertas familias de discos duros tienen una cabecera en los módulos, `0x52 0x4f 0x59 0x4c` [hexadecimal para el ASCII *ROYL*], que no tienen los primeros; la arquitectura *ROYL 20B* presenta la cabecera *ROYL* en sus módulos, pero tiene cambios estructurales en el área de servicio frente la arquitectura ROYL, además de presentar el módulo con identificador 0x20B, del que no dispone la anterior [53] [54].

Para averiguar los parámetros de geometría del disco duro necesarios, tenemos que acceder a las siguientes páginas de configuración, y leer los valores que se pueden encontrar en las siguientes posiciones (ver Tabla 2, Tabla 3 y Tabla 4), en cada caso [Anexo 1]:

Arquitectura Marvell	
Parámetro	Se obtiene de la página de configuración
SPT en el área de servicio (valor dependiente de la superficie)	0x05 (offset 0x28, tamaño 0x02 por cabezal)
Cilindros en el área de servicio	0x01 (offset 0x24, tamaño 0x02)
Cabezales activos	0x01 (offset 0x1E, tamaño 0x01)

Tabla 2: *Ubicación de los parámetros de geometría en familias Marvell*. Elaboración propia.

Arquitectura Marvell ROYL	
Parámetro	Se obtiene de la página de configuración
SPT en el área de servicio (valor dependiente de la superficie)	0x05 (offset 0x6C, tamaño 0x02 por cabezal)
Cilindros en el área de servicio	0x01 (offset 0x24, tamaño 0x02)
Cabezales activos	0x01 (offset 0x1E, tamaño 0x01)

Tabla 3: *Ubicación de los parámetros de geometría en familias Marvell ROYL*. Elaboración propia.

Arquitectura Marvell ROYL 20B	
Parámetro	Se obtiene de la página de configuración
SPT en el área de servicio (valor dependiente de la superficie)	0x05 (offset 0x6C, tamaño 0x02 por cabezal)
Cilindros en el área de servicio	0x01 (offset 0x24, tamaño 0x02)
Cabezales activos	0x01 (offset 0x1E, tamaño 0x01)

Tabla 4: *Ubicación de los parámetros de geometría en familias Marvell ROY 20B*. Elaboración propia.

Para obtener las páginas de configuración, utilizamos la siguiente página de log, que empaqueta el comando VSC `0x000d`:

```
word 0: Para acceso a páginas de configuración usamos el
VSC 0x000d [55]
```

```
word 1: página de configuración L (0x0001, 0x0005)
word 2: página de configuración H (0x0000)
words 3-256: padding (0x00)
```

Todas las palabras de 2 bytes han de escribirse LE en este caso también, por lo que la cadena escrita para leer la página de configuración 0x05 es:

```
0x0d 0x00 0x05 0x00 0x00 0x00 0x00 … padding
```

3.3.5. Cálculo del tamaño del espacio disponible en el área de servicio

Conociendo los métodos para extraer la geometría del área de servicio del disco duro y los métodos para leer y escribir en esta zona, ya tenemos casi toda la información que necesitamos para saltar al siguiente nivel, que es el del desarrollo del driver. Aunque esta información es suficiente para escribir un driver que aprovecha en gran medida el espacio disponible en el área de sistema, se puede afinar más el cálculo del espacio disponible, teniendo en cuenta el espacio no usado de las superficies de sistema principales (las que guardan la copia 0 y la copia 1). Para ello, hemos de analizar el directorio de módulos, para saber cuánto ocupan y dónde están en el área de sistema los módulos que usa el disco duro para funcionar, y poder así discriminar los sectores en desuso de estas superficies.

Examinando el directorio de módulos en superficie de un disco duro Western Digital se puede observar:

1. Que se trata de una tabla (a partir de cierto byte);
2. Que en ella se pueden identificar direcciones de módulos y extensión;
3. El tamaño total, en sectores (o en bytes) que ocupa el área de sistema en las superficies.

Estas observaciones se han podido realizar mediante una sencilla técnica de identificación de patrones y con ayuda de algunas de las herramientas enumeradas en el Capítulo 2 para validar las conclusiones a las que se llega. Se detalla a continuación

(ver Figura 12) la estructura de un directorio de módulos extraído de un disco duro Western Digital de la familia Marvell ROYL 20B, *Shrek*:

Figura 11: *Directorio de módulos en un disco duro de la familia Shrek*. Elaboración propia.

En el bloque anterior se identifican los siguientes fragmentos de información:

- `0x00000000`: **`0x524F594C`** = "ROYL", es la cabecera que arbitrariamente identifica estos módulos (familia *Marvell ROYL*).
- `0x00000008`: **`0x0100`** = ID del módulo, en LE = 0x0001. Es el identificador del módulo; en este caso es el módulo 1, que es el directorio de módulos.
- `0x0000000A`: **`0x2000`** = Tamaño del módulo en sectores. Este módulo se extiende a lo largo de 32 sectores (nótese que los campos enteros se almacenan todos en *little endian*).
- `0x0000000C`: **`0x26B0F713`** = Código de comprobación de integridad del módulo (*checksum* o *CRC*). Cambiar este valor provoca que el disco duro no identifique un módulo por válido, y que no finalice su secuencia de arranque

correctamente (el disco duro reportará una condición de error activando los registros de error)[8].

- ➢ `0x00000010:` **`0x3030303230303030`** = 00020000; es la versión del módulo codificada en ASCII.
- ➢ `0x0000004C`: Aquí comienza la tabla propiamente dicha, que se agrupa en filas a continuación (se resalta en verde el tamaño del módulo en sectores):

```
1A 04 35 00 0A 00 0F 18 80 00 20 00 00 00 20 00 00 00 3B 07 00 00 E2 44 00 00
1A 02 2E 01 2A 00 03 18 00 00 AC BA 04 00 AC BA 04 00 00 00 00 00 00 00 00 00
1A 02 02 01 01 00 03 18 10 00 2A 00 00 00 2A 00 00 00 00 00 00 00 00 00 00 00
1A 02 03 01 04 00 03 18 10 00 2B 00 00 00 2B 00 00 00 00 00 00 00 00 00 00 00
1A 02 04 01 01 00 03 18 10 00 2F 00 00 00 2F 00 00 00 00 00 00 00 00 00 00 00
1A 02 05 01 02 00 03 18 10 00 30 00 00 00 30 00 00 00 00 00 00 00 00 00 00 00
1A 02 06 01 02 00 03 18 10 00 32 00 00 00 32 00 00 00 00 00 00 00 00 00 00 00
1A 02 08 01 25 00 03 98 10 00 67 00 00 00 67 00 00 00 00 00 00 00 00 00 00 00
1A 04 02 00 05 00 0F 18 90 02 A5 DB 04 00 A5 DB 04 00 20 00 00 00 20 00 00 00
1A 04 1B 00 7E 00 0F 18 90 00 AA DB 04 00 AA DB 04 00 25 00 00 00 25 00 00 00
1A 04 11 04 E3 00 0F 18 90 02 28 DC 04 00 28 DC 04 00 A3 00 00 00 A3 00 00 00
…
```

Cada fila representa una entrada de un módulo en la tabla. Cada entrada tiene la siguiente estructura:

- Offset 0x00, tamaño 1: Tamaño de la entrada. Todas las que se muestran tienen tamaño 0x1A = 26.
- Offset 0x02, tamaño 2: ID del módulo, LE.
- Offset 0x04, tamaño 2: Longitud del módulo en sectores de 512 bytes, LE.
- Offset 0x0A, tamaño 4: Dirección ABA en que comienza el módulo, LE.

Con esta información se puede calcular el tamaño en sectores de todos lo módulos referenciados en la tabla. Para la tabla del ejemplo, se obtiene un tamaño de 289614 sectores, equivalente a 144807 kB (~144 MB), repartidos en 504 módulos. Para

[8] El estándar ATA/ATAPI 7, volume 1, rev. 4b; Error Register (página 57) no prevé el establecimiento del registro de error sin la ejecución de un comando por parte del *host*; no obstante se ha observado que fabricantes como Western DIgital o Hitachi activan este registro cuando encuentran una condición de error durante el arranque.

calcularlo, se ha desarrollado un pequeño programa en Java capaz de leer todas las entradas listadas, todas con tamaño 0x1A [Anexo 2].

Una vez conocido el mecanismo con el que nos podemos comunicar con el área reservada de los discos duros (en concreto, del fabricante Western Digital, familia Marvell), podemos comenzar con el desarrollo de un controlador para Linux que nos permita acceder a esta zona desde el sistema operativo, como a cualquier otro dispositivo de bloques.

Capítulo 4

Desarrollo del controlador de disco duro capaz de acceder al área de servicio y aprovechar el espacio disponible

4.1. Introducción

Un controlador de disco es un software crucial en cualquier sistema operativo de gran alcance, al ser el programa o conjunto de programas que controlan a bajo nivel el acceso a diferentes sistemas de almacenamiento: discos duros, *pendrives*, tarjetas de memoria, disquetes, etc. En el caso de un sistema operativo de gran calado, como Windows, Mac OS o Linux, se trata una pieza de código diseñada con especial cuidado, ya que sobre ella se va a depositar una gran confianza, teniendo que cargar con la responsabilidad de salvaguardar uno de los activos más importantes, si no el más importantes, de cualquier empresa o usuario del siglo XXI: la información. Es una pieza de código diseñada conforme a un riguroso estándar, actualmente el ATA/ATAPI-8, que dicta el intercambio de señales, modos de operación, control de errores, y en general, gobierna el protocolo de comunicaciones que se exigen mutuamente el disco duro y el sistema operativo para funcionar e intercambiar información.

Para el desarrollo del controlador planteado, surge no obstante un problema: no existe estándar para comunicarse con las zonas reservadas de los discos duros. Este hecho, junto con el hecho de que cada fabricante se reserva sus propios mecanismos privados para comunicarse con estas áreas cuando el disco queda preparado para su venta, plantea varios problemas:

a. No se puede diseñar un controlador universal e inconsciente del modelo o fabricante que desarrolla el disco duro, lo cual es incompatible con el estándar ATA/ATAPI.

b. No se puede disponer de una garantía de su correcto funcionamiento. Los fabricantes son muy reacios a publicar información sobre el funcionamiento puertas a dentro de sus dispositivos, y nada garantiza que puedan quedar dañados o se pueda perder información como resultado de tratar de comunicarse con sus áreas reservadas.
c. Desde el punto de vista forense, no se puede certificar que el envío de comandos de carácter privado al disco duro no vaya a alterar el estado del disco duro, ni la información que el mismo contiene en algún modo. No obstante, la repetibilidad de las operaciones realizadas con el disco desde esta perspectiva puede ayudar en este aspecto.

4.2. Elección de un sistema operativo para el desarrollo del controlador

Aunque el controlador a diseñar sigue un principio de funcionamiento extensible a cualquier sistema operativo, se ha escogido un sistema operativo Linux genérico como base para construirlo, por los siguientes motivos:

a. Los controladores de disco duro forman parte de una pieza vital del sistema operativo, al tratarse de un *kernel-mode driver*[9], en lenguaje de Microsoft. Los *drivers* son piezas de software con acceso a muy bajo nivel de los recursos hardware de un sistema, y son potencialmente peligrosos y dañinos si no están bien diseñados. Es por ello que, sistemas operativos como Windows (en sus versiones de 64 bit de Windows Vista y posteriores), requieran validar mediante certificados digitales la procedencia de un *driver* para permitir que sea usado por el sistema [56].
b. Existen múltiples recursos de ejemplo en Internet sobre cómo escribir un controlador genérico para dispositivos de bloques, que sirven como punto de partida para el desarrollo de este controlador.

[9] El *kernel* o *ring 0 mode* hace referencia al anillo de protección con más privilegios de un sistema, con acceso indiscriminado a todos los recursos disponibles.

c. El desarrollo de controladores no está fuertemente ligado a un entorno y unas herramientas concretas en Linux, al contrario que en Windows (Microsoft Visual Studio) o Mac OS (XCode).
d. Y, particularmente, tengo más experiencia desarrollando para Linux que para cualquier otra plataforma, por lo que puedo empezar el desarrollo antes y aprovechando conocimientos de los que ya dispongo.

Dentro de las posibilidades que ofrece Linux, el desarrollo del controlador tendrá por objetivo ser funcional en versiones del *kernel* de Linux posteriores o iguales a la 2.6. Esto es así debido a considerables cambios y mejoras que se incluyeron en el kernel de Linux en lo concerniente al API del kernel y el interfaz de dispositivos de bloque o *block devices* [57], además de la disponibilidad de recursos en Internet con ejemplos de implementación orientados a estas versiones del *kernel* de Linux.

4.3. Desarrollo del controlador

4.3.1. Requisitos previos y medios utilizados

El desarrollo del controlador se ha realizado en el lenguaje de programación C. Se ha llevado a cabo en un ordenador con una distribución de Linux instalada, Debian [58], en su versión 7 de 32 bits, y con *kernel* 3.2.0-4-686-pae. El procesador de textos usado es *nano* [59]. Se ha utilizado el siguiente equipo para realizar este desarrollo:

- Placa base ASRock P5B-DE, con 2 puertos SATA integrados (4 *slots* SATA).
- CPU Intel Core™2 Quad Q6600 a 2.40 GHz
- 8 GB de memoria RAM
- Disco duro para hacer pruebas: Western Digital modelo WD2500JS-22NCB1 de 250 GB

El disco duro sobre el que se han hecho todas las pruebas es un Western Digital de la familia Marvell, modelo WD2500JS, de 250 GB de capacidad. El disco duro no contenía información importante.

El controlador a desarrollar necesita implementar ciertas interfaces definidas en el *kernel* de Linux, por lo que además de las librerías estándar de C, es necesario instalar las cabeceras del *kernel* instalado, conocidas como las *kernel headers*.

Por último, para compilar el código se hace uso de los *toolchains* que vienen por defecto con el sistema Debian, y el programa *make* [60].

4.3.2. Partiendo de un controlador de disco básico (*RAM Disk*)

Para el desarrollo del controlador partimos de un ejemplo básico que no realiza accesos a puertos de E/S, sino que emula un dispositivo de almacenamiento por bloques en memoria RAM (un *RAM drive*) [61], y que implementa las funciones e interfaces básicas [62]:

a. Registro del controlador en el *kernel* de Linux. Esto se consigue mediante la llamada a la función *`register_blkdev(unsigned int major, const char* name)`*, declarada en <linux/fs.h>, y obtención del *major number*.
b. Cancelación del registro del controlador en el *kernel*, mediante la llamada a la función *`unregister_blkdev(unsigned int major, const char* name)`*, declarada en <linux/fs.h>
c. Declaración de la estructura que define el dispositivo. Esta estructura contiene como mínimo información sobre el tamaño de dispositivo, un *spinlock* para garantizar que el recurso solo es accedido por un proceso a la vez, y una estructura de *gendisk*, que es la representación interna de nuestro dispositivo en el *kernel* de Linux, según se declara en <linux/fs.h>.

   Esta estructura, en su forma más elemental, es como sigue:

```
static struct sbd_device {
     unsigned long size;
     spinlock_t lock;
     struct gendisk *gd;
} Device;
```

d. Registro del *gendisk* en el kernel, con los *minor numbers* que se quieran dedicar al mismo (representación de la cantidad de particiones que puede manejar el controlador para este dispositivo), e inicialización de la estructura de *gendisk*.

   El registro del *gendisk* en el kernel se realiza como sigue (*minor number* = *16*):

```
Device.gd = alloc_disk(16);
if (!Device.gd) {
     goto out_unregister;
}
```

   Es necesario comprobar que el *kernel* ha permitido el registro del controlador, y abortar en caso de fallo. Si se ha podido registrar la estructura de *gendisk*, se continúa con su inicialización:

```
Device.gd->major = major_num;
…
strcpy (Device.gd->disk_name, "afd");
set_capacity(Device.gd,
     nsectors * (hardsect_size/KERNEL_SECTOR_SIZE));
```

   El cálculo del valor de `nsectors` se discute más adelante, en la integración con el acceso al disco duro.

   El nombre ***afd*** es el nombre dado al controlador, de *antiforensics block device*. Es el nombre con el que se mostrarán estos dispositivos al sistema operativo, para diferenciarlos rápidamente de los habituales *hdx* y *sdx* [63].

   El controlador, comenzando con *kernels* 2.6, debe crear y configurar explícitamente una cola de peticiones, o *request queue*, y registrarla en el *gendisk*:

```
Queue = blk_init_queue(afd_request, &Device.lock);
…
blk_queue_logical_block_size(Queue,
logical_block_size);
```

Al inicializar la cola de peticiones, se pasa una referencia a la función *afd_request*, que es la que se encargará de manejar cada petición que llegue al controlador, ya sea de lectura o de escritura [64].

e. Para que nuestro dispositivo admita particionado y ser formateado por herramientas como **fdisk** [65], es preciso que presente algún tipo de geometría al sistema. Para ello, declaramos la operación getgeo en el bloque de operaciones que admite el controlador:

```
int afd_getgeo(struct block_device * block_device,
struct hd_geometry * geo) {
    …
}
```

A través de esta función le informaremos al *kernel* sobre la cantidad de cilindros, cabezas y sectores de que dispone el disco duro, una vez obtenidos.

f. Una vez se ha inicializado el *gendisk* con la cola de peticiones y el resto de parámetros, lo único que resta es añadir el dispositivo al sistema. Para ello:

```
add_disk(Device.gd);
```

El controlador de *RAM drive* usado como base inicializa todas estas estructuras en su secuencia de arranque, y reserva un espacio en memoria RAM que actúa como disco duro volátil. Este driver conoce por tanto de antemano la información que necesita para funcionar y presentar el dispositivo al sistema:

- ➢ El tamaño del dispositivo (el tamaño del bloque de memoria RAM que se decida reservar para este uso).
- ➢ La geometría del dispositivo de almacenamiento. Al tratarse de un bloque de memoria RAM, no hay realmente una geometría que tratar o traducir.
- ➢ El puerto al que dirigir el flujo de información. No es un controlador que realmente haga operaciones de E/S con dispositivos externos, por lo que esta información no es algo que necesite averiguar para funcionar.

Estos tres puntos son los que se deben tratar en un controlador que pretenda comunicarse con un dispositivo de almacenamiento real, y se tratan en el punto siguiente.

La secuencia completa de inicialización del *driver* utilizada se puede consultar en el código anexo a esta memoria: ***afbd.c***.

4.3.3. Agregando las funcionalidades de acceso a las áreas reservadas

Nuestro controlador sabe registrarse en el sistema, pero aún no sabe identificar un disco duro conectado a la máquina, ni sabe si puede trabajar con su área de sistema, ni el espacio disponible en la misma. Se exponen a continuación las diferentes funcionalidades que se han implementado para el funcionamiento del controlador.

4.3.3.1. Identificando el dispositivo conectado

El controlador trata de encontrar dispositivos durante su inicialización, para registrarlos en el sistema operativo. La secuencia de arranque debe, por tanto, ser consciente de la posibilidad de que existan discos duros enchufados a un puerto de E/S. Esto lo hace automáticamente el controlador al quedar instalado en el sistema, o al insertarse mediante el comando de Linux *insmod* [module_name] [66], como sigue:

```
# insmod afbd.ko
```

El controlador diseñado no es capaz, en su versión actual, de escanear cada puerto de E/S en el sistema en busca de discos duros, y debe proporcionarse a priori la dirección del puerto (fijada en el código).

Conocido el puerto de E/S al que se ha conectado un disco duro, el driver implementa un método definido en el estándar ATA/ATAPI-8 y anteriores, que es el identificación de dispositivo (IDENTIFY DEVICE, comando ECh [67]), e imprime parte de la

identificación de dispositivo mediante un mensaje de kernel, llamando a la función *printk()*. Esto ha servido durante el desarrollo del controlador para tener la seguridad de que el puerto de E/S al que está accediendo es al que está conectado el disco duro al que queremos acceder, y no cualquier otro puerto de E/S.

Esta funcionalidad se ha implementado de la siguiente manera:

```
static int __init afd_init(void) {

      (…) //Inicialización

      struct ide_id id;
      get_dev_id(&id);
      char model[64];
      memset(model, 0, 64);
      memcpy(model, id.model, 40);
      flip(model, 40);
      printk (KERN_NOTICE "afd: Detected HDD: %s\n", model);

      (…) //Continúa con la inicialización
}
```

La estructura *ide_id* es un bloque de datos que almacena la respuesta del dispositivo al comando de la inicialización (según definido en estándar ATA/ATAPI, donde se indica que devuelve un bloque de 256 palabras [512 bytes], estando el modelo especificado en las palabras 27-46 [68]), con los campos (se indican en la estructura el bloque que contiene el modelo y el que tiene el número de serie):

```
static struct ide_id {
    short general_configuration;
    short obsolete1[9];
    char serial_number[20];    //Número de serie
    short obsolete2[3];
    char firmware_version[8];
    char hdd_model[40];        //Modelo del dispositivo
    …
};
```

La función `get_dev_id(&id)` lanza el comando **`ECh`** al disco duro tal como se define en el estándar ATA/ATAPI, siendo requerido que todo dispositivo que desarrollado bajo el estándar lo tenga implementado, por lo que podemos estar seguros de que vamos a ser capaces de identificar correctamente un disco duro sano.

Comprobamos también que se llama a una función `flip(model, 40)` antes de mostrar el modelo en la consola. Las convenciones adoptadas por el estándar ATA/ATAPI para la transmisión de cadenas de caracteres obliga a dar la vuelta a los caracteres recibidos de dos en dos, ya que la información se envía en palabras de 16 bits, en LE [69].

Con esta información podemos desarrollar un controlador mínimo, capaz de detectar el disco duro conectado en el puerto de E/S facilitado, y capaz de mostrar información sobre el mismo por la consola de depuración del *kernel* (ver Figura 12).

```
[   15.131252] ADDRCONF(NETDEV_UP): eth0: link is not ready
[  395.557462] afd: get_drive_id - STATUS BEFORE: BSY=0 DRD=1 DSC=1.
[  395.558334] afd: Detected device: WDC WD2500JS-22NCB1
[  395.558336] afd: dev_get_cp - STATUS BEFORE: BSY=0 DRD=1 DSC=1.
```

Figura 12: *Consola de depuración del kernel, mostrando el dispositivo detectado*. Elaboración propia.

4.3.3.2. Comprobando si el dispositivo detectado es compatible con el controlador

El controlador, en su versión actual, solo funciona con discos duros del fabricante Western Digital, y de unas familias en concreto (Western Digital Marvell). El primer paso durante la inicialización del *driver* y decidir si continuar o no es la correcta identificación del dispositivo, y la comprobación en una lista del modelo encontrado, para comprobar su compatibilidad. Se ha confeccionado una lista de modelos conocidos de discos duros Western Digital, junto con la familia a la que pertenecen [Anexo 3]. Cualquier otro disco duro detectado no debe dejar al controlador finalizar la inicialización con éxito, ya que los comandos específicos de fabricante enviados podrían tener efectos adversos en otros dispositivos.

4.3.3.3. Obteniendo y calculando los parámetros de geometría del disco

El controlador, en su fase de inicialización, debe ser capaz de extraer del disco duro sus parámetros geométricos, para poder calcular cuánto espacio hay disponible, y poder así terminar la inicialización y reportar al *kernel* las características del dispositivo encontrado.

Los parámetros a determinar en esta fase son los siguientes:

- Cabezales disponibles en el disco duro
- Mapa de cabezales o *headmap*
- Cilindros reservados para el área de servicio
- Sectores por pista en el área de servicio (SPT)
- Opcionalmente, se puede calcular el tamaño total que ocupan los módulos del área de servicio, para aprovechar todo el espacio disponible[10].

Afortunadamente, disponemos del conocimiento necesario para implementar funciones que nos ayuden a obtener esta información (Capítulo 3).

Obtención de los cabezales disponibles en el disco duro

En todos los modelos observados, esta información se ha encontrado en la página de configuración 0x01 [Anexo 1]. El lugar exacto depende de la familia de Western Digital, según se muestra en la siguiente tabla (ver Tabla 5):

[10] No se ha implementado este apartado en el controlador en su versión actual. Esta técnica, descrita en el Capítulo 3, permite aprovechar todo el espacio disponible, incluyendo el de las superficies de sistema (las que contienen copias del firmware en uso).

Familia	Ubicación del número de cabezales disponibles en CP 0x01	Tamaño (bytes)
WD Marvell	0x1E	1
WD Marvell ROYL	0x1E	1
WD Marvell ROYL 20B	0x1E	1

Tabla 5: *Ubicación del parámetro* `virtual heads` *por familia*. Elaboración propia.

El parámetro que es conveniente consultar es que de cabezales *virtuales*, es decir, no solo los que están instalados, sino los que están activos. Es importante ya que hay discos duros con cabezales instalados pero desactivados en el firmware estático de la memoria ROM, y por tanto inoperables.

Para la obtención de este parámetro, se define una función:

```
static short dev_get_vheads(char** cp1);
```

Que recibe como parámetro un puntero indirecto a la página de configuración 0x01 ya leída del disco. La extracción del parámetro es directa:

```
short vheads;
memcpy(&vheads, cp1 + 0x1E, 2);
return(vheads);
```

La lectura de la página de configuración se realiza mediante la función:

```
static int dev_get_cp(short cp, char** out_buff, int* len);
```

Siendo cp = 0x01 para la obtención de la página de configuración 0x01, y el resto de parámetros simplemente punteros cuyo contenido se establece dentro de la función, indicando el tamaño de la página de configuración leída, y su contenido. El funcionamiento de esta función queda descrito en el Capítulo 3, mediante el uso de un VSC.

Obtención del mapa de cabezales del disco duro

La mayoría de discos duros examinados tienen el firmware en platos magnéticos en las superficies físicas 0 y 1 (primer plato magnético). No obstante se han observado excepciones (discos duros sin superficies físicas 0 y/o 1 activas), por lo que no se puede dar por supuesto, y es imprescindible tener en cuenta esta posibilidad. Para ello, consultamos el parámetro *headmask*, que nos ofrece una diapositiva de qué cabezales hay activos y qué cabezales hay desactivados, en el bloque completo de cabezales (ver Tabla 6):

Familia	Ubicación de la máscara de cabezales activos en CP 0x01	Tamaño (bytes)
WD Marvell	0x1F	1
WD Marvell ROYL	0x1F	1
WD Marvell ROYL 20B	0x1F	1

Tabla 6: *Ubicación del parámetro* headmask *por familia*. Elaboración propia.

El parámetro se puede extraer con la misma técnica que se extrae el número de cabezales activos. Se define para ello la función:

```
static short dev_get_headmask(char** cp1);
```

Con ayuda de este parámetro, podemos crear una estructura equivalente al bloque de cabezales activos del disco duro, estableciendo los campos (valor 1) o vaciándolos (valor 0) en función de si el cabezal está activo o no:

```
static struct headmap {
     short h0;
     short h1;
     …
     short h7;
}
```

Nótese que solo se contemplan 8 cabezales en esta estructura. Las páginas de configuración examinadas solo tienen huecos suficientes para definir 8 cabezales en varias de sus estructuras, por lo que cualquier disco moderno con más de 4 platos magnéticos, no está soportado por este controlador.

En la versión actual del controlador hace una suposición, por simplicidad, dando por hecho que las superficies 0 y 1 son las de sistema, y no se utilizan en el cálculo del tamaño disponible del área de servicio[11].

Obtención del número de cilindros reservados para el área de servicio

El siguiente parámetro a determinar es el número de cilindros *negativos* o cilindros reservados. Esta información se obtiene en todos los modelos observados de la página de configuración 0x01, según siguiente tabla (Tabla 4):

Familia	Ubicación de la máscara de cabezales activos en CP 0x01	Tamaño (bytes)
WD Marvell	0x24	2
WD Marvell ROYL	0x24	2
WD Marvell ROYL 20B	0x24	2

Tabla 7: *Ubicación del parámetro que indica el número de cilindros reservados por familia*. Elaboración propia.

Para la obtención de este parámetro, se define la función:

```
static int dev_get_sacyl(char** cp1);
```

La función devuelve un número indicando, por cada superficie, cuántos cilindros se han reservado para el área de servicio.

[11] El controlador no funcionará con discos duros que no tengan al menos tres superficies activas (una usable totalmente).

Obtención del número de sectores por pista en el área de servicio

Para calcular el tamaño exacto disponible en el área de sistema (salvo las superficies de sistema que contienen el firmware, que no utilizaremos), lo último que resta por conocer es el SPT o sectores por pista en el área de servicio. Este parámetro se puede obtener de la página de configuración 0x05, según la siguiente tabla (Tabla 8):

Familia	Ubicación de la máscara de cabezales activos en CP 0x05	Tamaño (bytes)
WD Marvell	0x28	2*
WD Marvell ROYL	0x6C	2*
WD Marvell ROYL 20B	0x6C	2*

Tabla 8: *Ubicación del parámetro* `SPT` *por familia*. Elaboración propia.

(*) El parámetro ocupa 2 bytes por cada cabezal activo.

Se define para la extracción de este parámetro la función:

```
static int dev_get_saspt(char** cp1, int royl);
```

Que devuelve un único número indicando el valor de SPT detectado para la superficie 0[12].

Obtención del número de sectores totales a usar por el controlador

El cálculo de la cantidad total de sectores disponibles para su uso por el driver, queda por tanto como sigue (ver Fórmula 3):

[12] Por simplicidad, solo usaremos el primer valor de SPT. No se han observado discos duros con valores diferentes por superficie en el valor de STP en el área de sistema.

$$n_{sectors} = (h_{virtual} - 2) \times A_{spt} \times A_{cyl}$$ [Fórmula 3]

Siendo $h_{virtual}$ el número de cabezales virtuales disponibles, A_{spt} el número de sectores por pista en el área de sistema, y A_{cyl} el número de cilindros reservados para el área de sistema.

Estos parámetros se extraen durante la secuencia de inicialización del controlador y una vez detectado un disco compatible en el puerto establecido[13] (identificando parámetros con Fórmula 3):

```
nsectors = sacyl * saspt  * aheads;
Device.size = nsectors * block_size;
```

El valor de `block_size` está actualmente fijado en 512 bytes y coincide con el tamaño del bloque que necesita el *kernel* para funcionar.

4.3.3.4. Trabajando con el área de sistema: `afd_request()`.

Una vez inicializado el controlador con el dispositivo encontrado y todas las estructuras que requiere el *kernel* de Linux, podemos empezar a trabajar con el disco duro. La interfaz de acceso a dispositivo de bloques nos obliga a implementar la función `afd_request(struct request_queue* q)` para manejar cada petición que haga el controlador al dispositivo:

```
static void afd_request(struct request_queue * q) {
    struct request * req;
    req = blk_fetch_request(q);

    while (req != NULL) {
        if (req==NULL || (req->cmd_type != REQ_TYPE_FS)) {
```

[13] Se sustrae 1 al número de cilindros por pista detectados debido a que se ha encontrado que en algunos discos duros el último cilindro está sin formatear (el comando ATA/ATAPI *READ VERIFY SECTOR* devuelve error).

```
                printk (KERN_NOTICE "Skip non-CMD request\n");
                __blk_end_request_all(req, -EIO);
                continue;
            }
            afd_transfer(&Device, blk_rq_pos(req),
    blk_rq_cur_sectors(req), req->buffer, rq_data_dir(req));
            if (! __blk_end_request_cur(req, 0) ) {
                req = blk_fetch_request(q);
            }
        }
    }
```

La implementación de esta función tiene una llamada a `afd_transfer(...)`, que es la encargada de discernir entre las peticiones de escritura y las de lectura. Recibe como parámetro:

a. Un puntero a la estructura que define el dispositivo de bloques
b. El primer sector a escribir (en coordenadas unidimensionales)
c. El número de sectores a escribir
d. Un puntero al bloque de datos que se desea escribir, o en el que se desea almacenar la información a leer
e. Un *flag* que indica si la operación es de lectura o de escritura

Esta función define el formato en que debemos indicarle al sistema la dirección a la que afecta nuestra operación de E/S: una coordenada unidimensional, o un sector en coordenadas LBA. Actualmente solo sabemos, según se detalla en el Capítulo 3, leer o escribir en el área de servicio preparando una página de log en la que se indica el sector, el cilindro y la cabeza en la que se quiere escribir. Esto nos obliga a implementar una función de traducción, o *translator* específico para comunicarnos con el área de servicio en coordenadas unidimensionales.

Diseño del traductor CHS a LBA (el translator*)*

Se define una estructura para trabajar con direcciones físicas en coordenadas tridimensionales (CHS), que necesitamos para saber dónde tenemos que escribir en o leer del disco duro con exactitud:

```
static struct pchs {
    int c;
    short h;
    short s;
};
```

Se define asimismo un mecanismo de traducción entre el LBA requerido, que es el dato con el que trabaja el kernel de Linux, y el CHS que necesitamos:

```
static struct pchs xlate_pba(sector_t _pba) {
    struct pchs chs;
    int pba = (int)(_pba);
    chs.c = -sacyl + (pba / (saspt * aheads));
    chs.h = (short)(((pba / saspt) % aheads) + 2);
    chs.s = (short)((pba % saspt) + 1);
    return chs;
}
```

El mecanismo de traducción traduce coordenadas unidimensionales (sectores en formato LBA) a coordenadas tridimensionales (CHS). Es una parte vital del controlador, dado que según el análisis realizado en el Capítulo 3, al disco duro le tenemos que indicar la zona del área de servicio que queremos escribir o leer indicando el cilindro, el cabezal y el sector, en lugar de simplemente transmitirle la dirección LBA como durante la operación normal de un disco duro actual. Está basado en el que se utilizaba antiguamente, en discos duros que aún trabajan en coordenadas CHS [70]:

$$C = \frac{LBA}{HPC \times SPT} \qquad \text{[Fórmula 4]}$$

$$H = mod\left(\frac{LBA}{SPT}, HPC\right) \qquad \text{[Fórmula 5]}$$

$$S = mod(LBA, SPT) + 1 \qquad \text{[Fórmula 6]}$$

Por último, definimos dos métodos, `read_dev_pchs` y `write_dev_pchs`, que hacen uso de todo lo anterior, y utilizamos en la función `afd_transfer(…)` según necesitemos leer del disco duro, o escribir en el mismo (extracto modificado para clarificar las fases en la comunicación):

```
static int read_dev_pchs (char* buffer, short head, int
        track, short sector, short scount) {

    vsc_mode_on();
    dev_smart_write_cmd();
    send_longvsc_cmd(0x000c, 0x0001, (track & 0xffff),
(track >> 16) & 0xffff, head, sector, scount);

    dev_smart_read_cmd(1);
    memset(a, 0, logical_block_size);
    read_buffer(a);
}
```

4.3.3.5. Registros de estado del disco duro

Las comunicaciones con el disco duro hacen uso de los 7 registros definidos en el estándar ATA/ATAPI, cuyas direcciones se calculan a partir de la dirección del puerto base. Los registros del disco duro tienen diferente significado si sobre ellos se realizan operaciones de lectura o de escritura, según la siguiente tabla (Tabla 9) [71]:

Dirección	Lectura	Escritura
P0 (puerto base)	N/A	N/A
P0 + 1	Registro de error	Registro de características
P0 + 2	Número de sectores	Número de sectores
P0 + 3	LBA (byte menos significativo)	LBA (byte menos significativo)
P0 + 4	LBA (byte medio)	LBA (byte medio)
P0 + 5	LBA (byte más significativo)	LBA (byte más significativo)
P0 + 6	Dispositivo	Dispositivo
P0 + 7	Registro de estatus	Comando (operación)

Tabla 9: *Registros ATA para la comunicación con el disco duro*. Elaborada con información del estándar ANSI INCITS 397-2005 ATA/ATAPI-7, p. 51.

Para la señalización, es de especial interés el registro de estatus, del cual se hace un uso extensivo. De este registro hacemos uso de varios *flags* o bits [72]:

7	6	5	4	3	2	1	0
BSY	DRDY	DF/SE	#	DRQ	Obsolete	Obsolete	ERR / CHK

Tabla 10: *Registro de estatus*. Extraído del estándar ANSI INCITS 397-2005 ATA/ATAPI-7, p. 51.

Nos son de especial interés los siguientes *flags*:

a. El flag *BSY*: indica que el dispositivo está realizando alguna operación, y no puede atender a más peticiones hasta que acabe (salvo lecturas de este registro de estatus).

b. El flag *DRQ*: indica que el disco está preparado para intercambiar información con el *host* o sistema al que está conectado. Una vez se ejecuta un comando (lectura, escritura), hay que esperar a que el disco esté preparado para el intercambio de información comprobando este *flag*.

c. Los *flags DRDY* y *DSC* (#). Indican conjuntamente que el disco duro está preparado para recibir comandos del *host*. Antes de enviar nuevos comandos al disco duro, comprobamos que ambos *flags* están levantados (su bit correspondiente establecido a 1).

En el controlador, definimos la función `read_regs()` que lee los contenidos del registro de estatus, y actualiza las variables de estado:

```
static void read_regs();
```

Las variables de estado se usan extensamente en el controlador para establecer tiempos de espera y comprobar que el disco duro está en el estado adecuado antes y después de lanzar cada comando.

4.3.3.6. Integración de todos los componentes y compilación del módulo

Todas las funciones necesarias se han implementado acorde con el análisis de funcionamiento realizado, y se han integrado en el controlador básico. Para la compilación del módulo y el uso del controlador en un entorno real, consultar el Anexo 4.

4.5. Pruebas realizadas

El controlador se ha compilado y utilizado en los siguientes escenarios:

a. Acceso al dispositivo *afd* directamente: escritura y lectura de sectores individuales.
b. Acceso al dispositivo *afd* directamente: escritura y lectura de sectores en cadena.
c. Formateo del dispositivo en sistema de ficheros vFAT, sin tabla de particiones (partición comienza en sector 0).

d. Uso del sistema de ficheros formateado de vFAT: montaje y desmontaje, creación de ficheros de texto y directorios en raíz, y ficheros de texto anidados.
e. Uso del sistema de ficheros formateado de vFAT: borrado de ficheros y directorios.
f. Formateo del dispositivo en sistema de ficheros NTFS, sin tabla de particiones (partición comienza en sector 0).
g. Uso del sistema de ficheros formateado en NTFS: creación de ficheros de texto y directorios en raíz, y ficheros de texto anidados.

4.5.1. Método utilizado

Las pruebas se han realizado sobre un entorno real, con un disco duro real enchufado a un puerto SATA de un ordenador. El disco duro utilizado ha sido el siguiente (Tabla 11):

Modelo	WD2500JB
Familia	Hawk
Código de familia	NC
Capacidad en partición de usuario	250 GiB
Capacidad en partición de sistema (usable)	90 MB
SPT en SA	720
Cilindros en SA	64
Cabezales instalados	6
Cabezales disponibles	4

Tabla 11: Datos del disco duro utilizado en las pruebas.

Una vez averiguada la dirección del puerto base al que está conectado el disco duro, se ha compilado el código y se ha generado el módulo insertable o *driver*.

Se han realizado dos tipos de pruebas: pruebas de funcionamiento y pruebas de rendimiento.

4.5.1.1. Pruebas de funcionamiento

El propósito de estas pruebas es comprobar que el controlador detecta el disco duro, es capaz de leer las páginas de configuración correspondientes, inicializarse correctamente y presentar al sistema un disco duro del tamaño adecuado. Una vez inicializado el controlador, se realizan también pruebas de funcionamiento de las funciones de lectura y de escritura.

El controlador se inserta en el *kernel* de Linux con el comando *insmod*:

```
# insmod afbd.ko
```

Inmediatamente después se comprueba el registro del sistema mediante el comando *dmesg*, obteniendo información sobre lo que ocurre durante la inicialización del controlador:

```
# dmesg
```

Los mensajes que se observan son resultado de una técnica de depuración utilizada, el *tracing* o depuración mediante la impresión de mensajes que contienen información relevante sobre la ejecución del programa [73]. En el código del controlador se observan numerosas llamadas a la función *kprint()*, definida en las cabeceras del *kernel* de Linux, y que imprime mensajes que se pueden observar en el registro que vemos al llamar a dmesg. Entre la información más interesante a mostrar para comprobar el funcionamiento del controlador, nos interesa:

a. El registro de estatus del disco duro en cada momento. En concreto, los *flags* BSY, DRD, DSC y DRQ.

b. El contenido del bloque de identificación del dispositivo, resultado de ejecutar el comando ECh (IDENTIFY DEVICE).
c. El contenido de las páginas de configuración que se solicitan al disco duro, para comprobar que el *parseo* de los campos de interés (SPT, etc.) es correcto.
d. El contenido del registro de error del dispositivo.
e. Nos interesa colocar *balizas* o puntos de control para saber que el *driver* está comportándose adecuadamente, y verificar que la implementación de las interfaces declaradas por el *kernel* de Linux es correcta.

En el momento en que se ha identificado el dispositivo correctamente y se han implementado las funciones de lectura y de escritura, se ha comenzado con pruebas de lectura, escritura, comprobación de integridad de ficheros, y formateo.

¿Es indetectable la información escrita en esta área?

Sin ayuda de este controlador, o de una herramienta específica capaz de acceder a estas zonas y leerlas, la información allí alojada pasa desapercibida para cualquier investigador forense que se apoye únicamente en herramientas tradicionales. Se ha realizado una prueba de clonado de disco duro con una herramienta hardware, VOOM HardCopy 3P[14] (ver Figura 14), con el siguiente *setup* (Figura 13):

[14] Herramienta hardware forense de clonado de discos duros. Detalles técnicos en la página web del fabricante, http://www.voomtech.com/#!hardcopy-3p/c1q2b

Figura 13: Esquema del setup usado para el clonado de un disco duro mediante una herramienta hardware

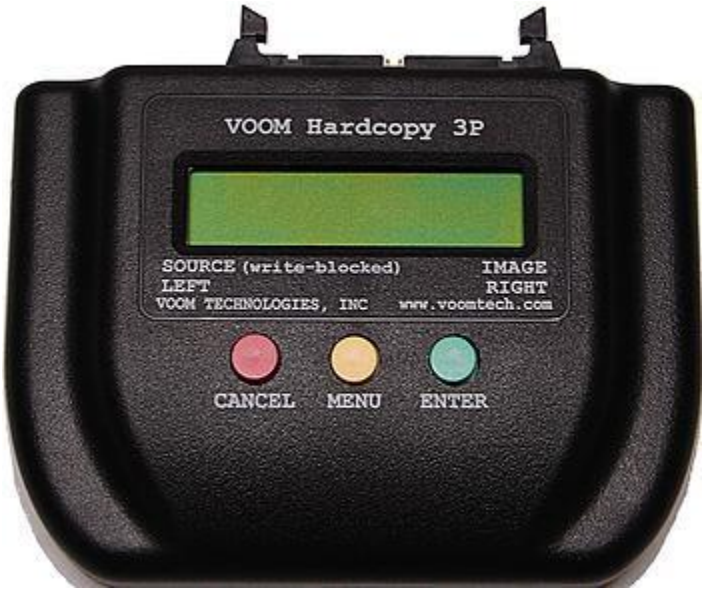

Figura 14: Clonadora de disco duro VOOM HardCopy 3P. Extraído de la web del fabricante, <http://www.voomtech.com/#!hardcopy-3p/c1q2b>

Esta herramienta hardware ha sido probada y validada por el NIST (Instituto Nacional de Estándares y Tecnología, por su sigla en inglés) [74], con buenos resultados,

haciendo referencia también a su capacidad para copiar sectores ocultos (mencionando en concreto los sectores ocultos mediante el uso de HPA).

Ambos discos duros, el de origen y el de destino, se han *saneado*[15] previo a las pruebas. Posteriormente se han escrito sectores en el espacio sobrante del área de sistema del disco duro de origen, y se ha procedido a realizar el clonado del mismo. El resultado ha sido satisfactorio para el controlador: tras examinar el disco duro de destino una vez realizado el clonado, se ha podido comprobar que **no se ha copiado la información escrita en el área de sistema del disco duro de origen**. Esto coincide con lo que se espera de una técnica de ocultación de información de este tipo, y se apoya en el siguiente razonamiento:

a. Para todas las máquinas de clonado de discos duros consultadas es suficiente para un clonado satisfactorio el conectar un disco duro de destino (el disco *target*) de igual tamaño al original (el disco *source*).

b. Las máquinas de clonado consultadas no indican sobre la necesidad de conectar discos duros del mismo fabricante y modelo; únicamente requieren que la capacidad del disco duro de destino sea igual (o superior) a la del disco de origen.

c. Dos discos duros aparentemente idénticos (mismo fabricante, mismo modelo, misma familia, misma serie, números de serie similares o consecutivos, y resto de datos identificativos iguales) no tienen por qué tener la misma configuración interna (número de cabezales activos, en primer lugar), y por tanto, el mismo espacio disponible en el área de servicio.

[15] El *saneamiento* de los discos duros, en terminología forense, se refiere al procedimiento de sobrescribir cualquier información que pudiera existir en un disco duro con información nueva, siguiendo un patrón fijo o aleatorio. En este caso, por facilidad en la identificación visual, se ha utilizado un patrón *nulo* (todos los bytes se escriben con el valor 0x00).

d. Si la máquina de clonado tuviese capacidad de copiar esta zona, no tendría lugar sobre el que copiarla. Tampoco se ha encontrado una máquina que solicite un dispositivo de almacenamiento adicional para volcar una copia de estas zonas.

e. Por lo tanto, estas máquinas de clonado de discos duros obvian las zonas reservadas de los discos duros.

4.5.1.2. Pruebas de rendimiento

El propósito de estas pruebas es medir la velocidad con la que se puede trabajar con el controlador. Se han realizado medidas de velocidad de lectura y de escritura.

Pruebas de lectura

Para la realización de estas pruebas se ha ejecutado un comando que lee del disco duro y descarta directamente los resultados (para que no influya el dispositivo de salida en la medida de la velocidad). El comando ejecutado ha sido el siguiente:

```
# dd if=/dev/afb of=/dev/null count=1024
```

El comando se ha lanzado repetidas veces, midiendo en cada iteración los resultados.

Pruebas de escritura

Estas pruebas han tomado como origen de datos un dispositivo virtual, el `/dev/zero`, que sirve como fuente de bytes nulos (0x00), y como destino el dispositivo que presenta nuestro controlador. El comando ejecutado ha sido el siguiente:

```
# dd if=/dev/zero of=/dev/afd count=1024
```

El comando se ha lanzado repetidas veces, midiendo en cada iteración los resultados.

4.5.2. Resultados

En las últimas pruebas realizadas, el controlador identifica correctamente el dispositivo y el tamaño disponible del área de servicio (Figura 15):

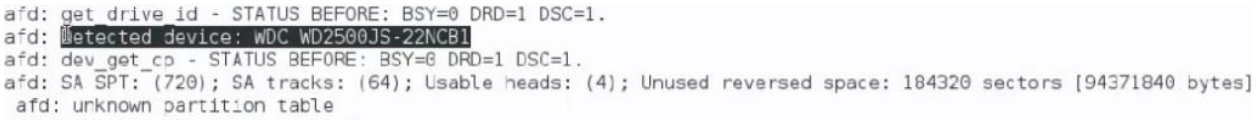

Figura 15: Detección del disco duro por parte del controlador.

Y presenta el área disponible como un nuevo dispositivo al sistema (Figura 16):

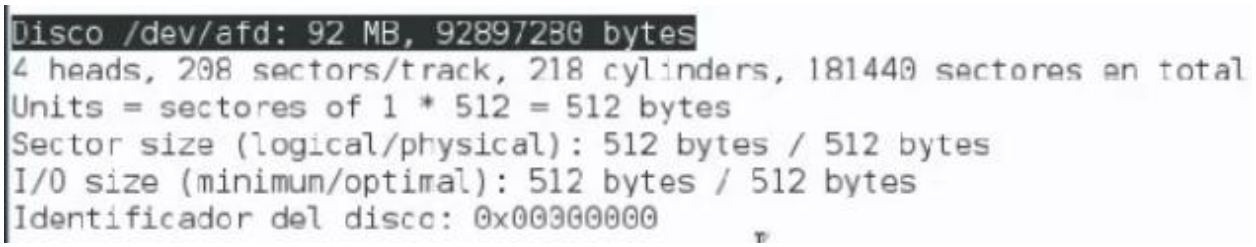

Figura 16: Detección de los parámetros de funcionamiento por parte del controlador.

Sobre el dispositivo presentado al sistema se han realizado diferentes pruebas de funcionamiento (lectura y escritura simple, diferentes tipos de formateo, ocultación efectiva de datos), obteniendo los siguientes resultados en cada caso (Tabla 12):

Prueba	Resultado	Comentarios
Lectura y escritura en sectores individuales	Correcto	Se utilizó la herramienta *hexedit* para el acceso y operaciones de E/S en dispositivo */dev/afd*.
Formateo en vFAT	Correcto	El sistema se ha formateado con la herramienta *mkfs.vfat*. El proceso es más lento de lo habitual (1 minuto). El sistema responde muy despacio mientras se está formateando.

Uso del sistema de ficheros vFAT (1)	Correcto	Se pueden escribir ficheros y directorios. La partición puede desmontarse y volver a montarse; la información sigue presente.
Uso del sistema de ficheros vFAT (2)	Error	El sistema se queda colgado eliminando el directorio creado. Al leer determinados ficheros, no se alcanza el final del mismo y se detecta información repetitiva (error de integridad de fichero).
Formateo en NTFS	Error	Se ha utilizado *mkfs*.ntfs. El sistema se queda colgado formateando el dispositivo.
Ocultación efectiva de información	Correcto	Las herramientas hardware de clonado de discos duros no son conscientes de la existencia de esta zona.

Tabla 12: Resultado de las pruebas de funcionamiento.

Las pruebas de rendimiento (lectura y escritura en cadena) obtuvieron los siguientes resultados:

Prueba	Resultado	Comentarios
Velocidad de lectura media	24,4 kB/s	En algunas iteraciones la velocidad ha caído hasta 17 kB/s.
Velocidad de escritura media	33,7 kB/s	Velocidad estable entre iteraciones.

Tabla 13: Resultado de las pruebas de rendimiento.

En general, el sistema se vuelve lento trabajando con el *driver* en su versión actual. Esto se ha considerado aceptable en su estado de desarrollo actual, dado que el controlador no está optimizado:

a. Los sectores se leen y se escriben de uno en uno. El estándar ATA/ATAPI define mecanismos de lectura y escritura en bloque (hasta 256 sectores de una vez).
b. No se implementa una cola de peticiones avanzada.

c. Cada petición que involucra el área de servicio requiere realizar más operaciones que una petición normal:
   i. Enviar *VSC Enable*.
   ii. Leer resultado (el comando ha terminado en error o no).
   iii. Enviar SMART WRITE log, con una dirección de log *vendor-specific*.
   iv. Leer el resultado del comando anterior.
   v. Enviar una página de log con el VSC (de lectura o escritura) cuando el disco está preparado para recibir información.
   vi. Leer el resultado del VSC.
   vii. Enviar SMART READ log, con una dirección de log *vendor-specific*.
   viii. Leer el resultado del comando.
   ix. Escribir o leer el bloque de datos (un único sector actualmente) en el puerto. Hacer esta operación tiene un coste alto: en las hojas de especificaciones de los discos duros el fabricante indica este coste como *read seek time*, y *write seek time*.
   x. Enviar *VSC Disable*.
d. El controlador tiene establecido el *flag REALLY_SLOW_IO*, que avisa al *kernel* de que se van a realizar operaciones de E/S lentas. Este *flag* se ha activado durante el desarrollo, en previsión de que el controlador no está optimizado y el *kernel* debe esperar tiempo suficiente entre cada petición.

Por otro lado, los errores detectados durante la eliminación de directorios, la comprobación de integridad de algunos ficheros, y el formateo en NTFS, apuntan a un problema en la traducción de direcciones[16], que debe investigarse y corregir.

[16] Se creó un fichero con varios párrafos de texto en su interior. Al abrirlo con la herramienta *gedit*, se puede leer el contenido, pero comienza a repetirse llegado un punto, y no se puede alcanzar el final del fichero.

4.5.3. Otros problemas

Se han detectado problemas de concurrencia a la hora de utilizar y realizar pruebas con el controlador, así como errores de lectura en algunos cilindros. Se detallan estos problemas a continuación.

4.5.1.1. Concurrencia

El controlador *afd*, en su forma actual, convive con el controlador integrado en el *kernel* de Linux, que muestra el dispositivo /dev/sda (Figura 17):

```
Disk /dev/sda: 250.1 GB, 250059350016 bytes
255 heads, 63 sectors/track, 30401 cylinders, 488397168 sectores en total
Units = sectores of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimun/optimal): 512 bytes / 512 bytes
Identificador del disco: 0x08c52ff7
```

Figura 17: El disco duro usado en las pruebas, detectado por el controlador del sistema

Ambos controladores acceden al mismo puerto de E/S de la controladora SATA integrada en la placa base del ordenador, y no son conscientes de la existencia del otro. Además, asumen que tienen acceso exclusivo al dispositivo, lo cual puede crear problemas: para las pruebas realizadas, se ha evitado en la medida de lo posible la comunicación del controlador integrado con el disco duro, habiendo desmontado las particiones en él presentes antes de realizar las pruebas. Hay que recordar que ambos dispositivos, aunque se muestran como diferentes, están en el mismo soporte físico; uno presenta la partición física del espacio de usuario, y otro la partición física del espacio de sistema o área de servicio.

4.5.1.2. Cilindros sin formatear

Varios de los modelos analizados presentaban errores de lectura (CRC) al tratar de leer sectores de los cilindros negativos en cualquiera de las superficies que no fueran las que tienen copias del firmware (superficies distintas a la 0 y la 1). Esto se debe no a que el medio esté defectuoso o el disco duro no tenga capacidad para leer estos sectores, sino que los mismos no está formateados, y el campo de ECC que tiene cada sector [75] no valida los datos leídos. Para solucionar este problema, hay que escribir estos sectores, escribiendo un campo de ECC nuevo, dejándolo correctamente formateado.

Se recomienda por tanto realizar un limpiado del área presentada por el controlador previo a su uso, para formatear todos los sectores previo a su uso y evitar errores de lectura y reintentos innecesarios de lectura por parte del disco duro.

Se adjunta un vídeo del funcionamiento y uso del driver como anexo de este proyecto, ***afbd.webm***[17].

[17] El video muestra cómo obtener la dirección del puerto donde se ha conectado el disco, la compilación del módulo y el uso del driver mediante pruebas de funcionamiento (escritura y lectura, formateo de un volumen).

Capítulo 5

Planificación y presupuesto

5.1. Planificación

La elaboración del proyecto se ha planificado en diferentes fases, agrupadas en cuatro grandes bloques propios de todo proyecto de I+D+i:

A. Investigación.

El bloque de investigación propiamente dicha identifica las siguientes tareas:

i. Estudio de viabilidad. Técnicas existentes en la actualidad que logran el mismo propósito, o propósitos similares.

ii. Análisis de las soluciones existentes. Estudio sobre las ventajas de nuestra solución frente a las existentes.

B. Desarrollo

Las fases identificadas en este bloque son:

iii. Desarrollo de una técnica de acceso a las áreas de sistema del disco duro.

iv. Implementación de un controlador básico de disco para Linux.

v. Integración de nuestra técnica de acceso a las áreas de sistema del disco duro en el controlador de disco.

vi. Implementación de un mecanismo de traducción entre coordenadas LBA y CHS para el almacenamiento lógico (secuencial) de la información.

C. Pruebas

Esta fase se divide en las siguientes tareas:

vii. Pruebas de funcionalidad.

viii. Corrección de errores.

ix. Pruebas de rendimiento.

D. Documentación

A lo largo de cada una de las tres fases anteriores se ha ido recogiendo información para la elaboración de esta memoria.

5.1.1. Diagrama de Gantt

A continuación se muestra el diagrama de Gantt elaborado para la planificación del proyecto de investigación y desarrollo del controlador (se adjunta anexo a este proyecto el fichero del proyecto, ***afbd.gan***):

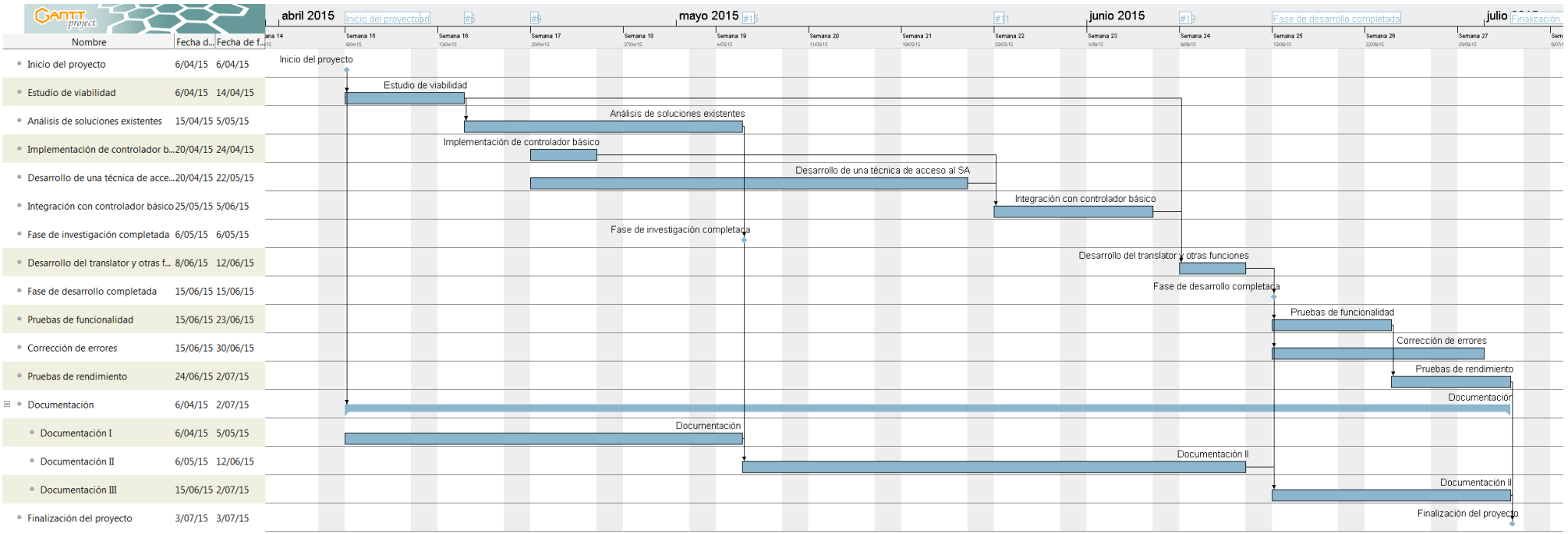

Figura 18: Planificación del proyecto: Diagrama de Gantt.

5.2. Presupuesto

La realización de este proyecto ha supuesto el uso de una combinación de recursos de hardware, de software y humanos, así como el uso de otros recursos de forma indirecta (luz y conexión a internet). El coste total se ha calculado sin tener en cuenta el IVA u otros impuestos trasladados al consumidor, y usando la siguiente metodología de estimación de presupuestos:

5.2.1. Recursos de hardware

En el cálculo del coste desprendido del uso de hardware que se ha hecho para la realización de este proyecto, se ha tenido en cuenta el coste de mercado del material utilizado, así como si la compra se ha realizado exclusivamente para la realización del proyecto, y en caso contrario, el tiempo de vida útil estimado de cada elemento hardware y el tiempo que se estima utilizarlo, según la sección 5.1.1. Se muestra en la siguiente tabla una relación de los elementos de hardware utilizados, junto con el coste calculado:

Elemento HW	Coste de mercado	Unidades	Tiempo de vida útil	Tiempo utilizado	Coste para el proyecto
Ordenador de sobremesa clónico	1650 €	1	6 años	3 meses	69 €
Discos duros para pruebas	50 €	3	-	-	150 €
Herramienta ACELab PC3000	5090 €	1	6 años	3 meses	212 €
Ordenador portátil para documentación	990 €	1	4 años	3 meses	62 €
Herramienta de clonado VOOM HardCopy 3P	1749 USD[18]	1	6 años	1 mes	22 €
Subtotal	**515 €**				

Tabla 14: Coste estimado de los recursos de hardware

5.2.2. Recursos de software

La siguiente tabla refleja el software que se ha utilizado para la elaboración del proyecto, en todas sus fases. Se incluye en esta tabla el coste de los documentos digitales que hubo que comprar para la fase de investigación. No se incluye el software *open source*:

[18] Se ha utilizado el cambio oficial a fecha 1 de junio de 2015: 1 EUR = 1,0927 USD. *Fuente: Bloomberg*

Nombre del software	Coste de mercado	Unidades	Tiempo de vida útil	Tiempo utilizado	Coste para el proyecto
ACELab PC3000 Utility	0 € (Incluida con el hardware)	1	6 años (La misma que el ordenador en que está instalado)	3 meses	**0 €**
Microsoft Office 365 Universitarios	65,29 €	1	Suscripción 4 años	3 meses	**4 €**
Documento digital Estándar BS ISO/IEC 27037:2012	158 CHF[19]	1	Comprado para proyecto	-	**151,80 €**
Subtotal	**155,80 €**				

Tabla 15: Coste estimado de los recursos de software

5.2.3. Recursos humanos

Al no estar regulado por ningún colegio profesional el coste por hora de los informáticos analistas y analistas programadores en España, se ha utilizado como referencia un expediente de contratación del Organismo de Informática del Ayuntamiento de Madrid, de servicios para el desarrollo de tareas de mantenimiento correctivo, adaptativo y evolutivo de diferentes sistemas de información del Ayuntamiento de Madrid [76], que incluye tablas de costes para diferentes perfiles informáticos. Se ha elaborado una tabla en base a estos costes, que se muestra a continuación:

[19] Se ha utilizado el cambio oficial a fecha 1 de abril de 2015: 1 EUR = 1,04086 CHF. *Fuente: Bloomberg*

Perfil	Coste por hora	Horas	Coste total
Analista	37,05 €	22 * 8	6520,80 €
Analista-programador	30,40 €	71 * 8	17267,20 €
Subtotal	**23788,0 €**		

Tabla 16: Coste estimado de los recursos humanos

5.2.4. Otros costes y gastos indirectos

La contabilización de costes indirectos incluye el gasto proporcional en el alquiler de la oficina, la factura de la luz y la conexión a Internet. Otros gastos como desplazamientos y dietas no se han incluido en el coste del proyecto, al no entenderse como derivados del mismo. El cálculo del porcentaje se ha realizado para un puesto de trabajo ubicado en una oficina de 75 m^2 compartida por 4 personas, cada una de ellas haciendo uso de una estación de trabajo de similares características.

Concepto	Coste total	Porcentaje achacable	Coste para el proyecto
Alquiler del local	650 €/mes	25%	487,50 €
Factura eléctrica	120 €/mes	25%	90 €
Factura Internet	29 €/mes	25%	21,75 €
Subtotal		**599,25 €**	

Tabla 17: Coste estimado de los gastos indirectos

5.2.5. Total

El coste total del proyecto se estima por tanto según la siguiente tabla, que recoge los puntos anteriores:

Concepto	Coste
Recursos de hardware	515
Recursos de software y documentos digitales	155,8
Recursos humanos	23788
Otros gastos	599,25
TOTAL	**25058,05 €**

Tabla 18: Coste total estimado del proyecto

Capítulo 6

Conclusiones

El estándar BS ISO/IEC 27037:2012 ofrece unas líneas generales de actuación para la identificación, recolección, adquisición y preservación de la evidencia digital. En sus primeras páginas ofrece unas definiciones de términos muy relevantes para la investigación, y que podrían alienar al investigador en la elección de los repositorios de información o evidencias digitales a investigar. En concreto, la definición 3.23 sobre el espacio no asignado o *unallocated space*:

> **Unallocated space**
> *Area on digital media, including primary memory, which has not been allocated by the operating system, and which is available for the storage of data, including metadata.*

Esta definición deja fuera del alcance del proceso de investigación forense las zonas reservadas del disco duro, pues las mismas no encajan con la descripción de *which is available for the storage of data* (que están disponibles para el almacenamiento de información), al ser un área no destinado al almacenamiento de datos de usuario, ni estar directamente disponible.

El controlador desarrollado da acceso de lectura y escritura a estas zonas reservadas, mejorando las capacidades de investigación de los equipos de análisis informático forenses, y capacitándolos para descubrir información oculta por individuos que ya pudieran conocer esta técnica con anterioridad, pero que no hubieran hecho público su uso.

El controlador diseñado, y otras técnicas similares, son prueba de que es preciso definir con mayor claridad qué son los repositorios de información, y de que los investigadores deben buscar información más allá de lo que pueda ver en un primer momento, especialmente tras ser conocido el perfil de las personas que utilizaban los medios a analizar, y si éste encaja con individuos que pudieran estar haciendo uso de

técnicas avanzadas de ocultación de información, más allá del cifrado de datos —que aunque difícil de deshacer, es relativamente sencillo de detectar cuando se trata de ocultar volúmenes grandes de información—.

Se llega a las siguientes conclusiones específicas:

i. El estándar BS ISO/IEC 27037:2012 no ofrecen una perspectiva completa de lo que debe buscar un examinador forense o DEFR en las fases iniciales de la investigación (identificación, recolección, adquisición, preservación).

ii. Los discos duros actuales disponen de zonas reservadas con hasta 1 GB de espacio disponible en los modelos observados. Estas zonas se pueden utilizar, realizando una investigación sobre la forma en que el fabricante del disco duro accede a las mismas, y desarrollando un controlador apropiado.

iii. Se puede ocultar información en estas zonas. Esta información no sería copiada por herramientas especializadas en el clonado de discos duros, ya sean software o hardware.

iv. Existen herramientas capaces de acceder a estas zonas, pero no tienen enfoque forense y su modo de operación es información privada, por lo que no se puede constatar su seguridad o fidelidad, y contravienen el requisito de reproducibilidad dictado por el estándar.

v. Es necesario hacer consciente a toda la comunidad forense de las posibilidades de ocultación de información existentes, como una que haga uso de un controlador similar al desarrollado, y fomentar el uso de herramientas como ésta, con las justificaciones necesarias sobre su uso en cada caso.

Capítulo 7

Líneas futuras de desarrollo

Tras el desarrollo del controlador y las pruebas realizadas, se han identificado varias vías de mejora y de optimización para el futuro:

i. Actualmente, el controlador solo es funcional con unas familias concretas de discos duros, de uno de los fabricantes que existen en el mercado (aun suponiendo una cuota del mercado de discos duros de alrededor de 30%). Por lo tanto, una línea trivial de continuación del desarrollo de este controlador es hacerlo extensible a más modelos y familias de discos duros, tratando de cubrir todo el espectro de posibilidades; para ello es necesario investigar las posibilidades de acceso a estas áreas reservadas que ofrecen el resto de fabricante, y analizar e implementar sus protocolos de comunicación específicos.

ii. El controlador actual está diseñado para funcionar únicamente en Linux, con versiones del *kernel* **k** tales que $2.6 \leq \mathbf{k} < 3.16$. Los cambios en las interfaces de acceso a dispositivos de bloque que ha sufrido el *kernel* en la versión 3.16 [77] obligan a realizar cambios en la forma en que se gestionan las peticiones de lectura/escritura. Se propone como línea de desarrollo actualizar el controlador para que sea compatible con las versiones más recientes del *kernel*, así como reescribirlo para otros sistemas operativos, como Windows o Mac OS.

iii. El controlador hace accesos individuales para cada sector a leer/escribir. Esta implementación es sub óptima, pudiendo empaquetar hasta 256 sectores en cada petición.

iv. El controlador no hace uso extensivo de una cola de peticiones: utiliza una cola simple, sin ninguna clase de gestión. Se puede estudiar el diseño de una cola para la gestión de peticiones, teniendo en cuenta el actual estado del arte.

v. En la versión actual, el controlador no accede a las superficies 0 y 1, asumiendo que estas superficies contienen el firmware activo del discos, y que cualquier error en el manejo de estas superficies puede suponer un daño irreparable en el dispositivo. Se propone una ampliación en el controlador, que durante la fase de inicialización calcule el espacio disponible en estas superficies y lo añada al espacio disponible total. Cabe comentar que esta ampliación complica considerablemente el diseño de las funciones de traducción LBA a CHS.

Cabe comentar en este aspecto que el auge de los discos duros de estado sólido en los últimos años amenaza la supervivencia a medio y largo plazo de los dispositivos de almacenamiento magnético, y que este controlador explota características muy concretas de los dispositivos de este tipo: platos magnéticos, cilindros sin utilizar, etc. Los discos duros de estado sólido no poseen esta serie de características, pero no dejan la puerta completamente cerrada: estos dispositivos tienen más capacidad de la que anuncian; el espacio adicional se reserva para nivelar el desgaste de las celdas de memoria [78], y es imprescindible una investigación para saber si los fabricantes de estos discos duros ofrecen alguna posibilidad para acceder a estas zonas mediante comandos fuera del estándar.

Adicionalmente, se propone el desarrollo de un controlador similar, que tenga por propósito únicamente el análisis (la lectura) de estas áreas y de forma completa, incluyendo las superficies 0 y 1, y que ofrezca un nivel adicional de seguridad y confianza durante las investigaciones forenses: un controlador que ignore las peticiones de escritura. Su implementación puede partir del código propuesto y entregado junto con esta memoria.

Anexo 1

Páginas de configuración en diferentes arquitecturas de Western Digital

La siguiente información es de elaboración propia, y se ha recopilado mediante técnicas de identificación de patrones sobre páginas de configuración leídas de discos duros cuya geometría ya era conocida (extraída mediante las herramientas mencionadas en el Capítulo 2). Los datos relevantes se representan en código de color, para identificar sus posiciones en las páginas de configuración correspondientes.

Arquitectura Marvell

Modelo de disco duro utilizado	WDC WD2500JS-22NCB1
SPT en área de servicio*	720 [0x02D0]
Cilindros en el área de servicio*	64 [0x0040]
RPM del disco duro*	7215 [0x1C2F]
Radios de servo (servo wedges)*	180 [0x00B4]
Zonas*	21 [0x0015]
Platos instalados en el disco duro*	3 [0x03]
Cabezales físicos de lectura/escritura*	6 [0x06]
Cabezales virtuales (lógicos) de lectura/escritura*	6 [0x06]
Máscara de cabezales activos	0b00111111 [0x3F]

Tabla 19: Ubicación de parámetros de interés en las páginas de configuración en discos duros WD familia Marvell

* Información reportada por la herramienta utilizada.

Página de configuración 0x01

```
0x0000 02 00 31 30 2E 39 54 45 00 00 30 32 2E 38 4B 00 ..10.9TE..02.8K.
0x0010 00 00 31 30 2E 39 54 45 00 00 63 16 03 06 06 3F ..10.9TE..c....?
0x0020 00 15 B4 00 40 00 2F 1C E0 47 01 00 14 32 44 05 ..´.@./.àG...2D.
0x0030 00 02 00 00 B2 31 00 00 B2 31 46 00 21 00 01 00 ....²1..²1F.!...
0x0040 04 00 00 00 00 00 01 16 2B 00 32 0A 00 01 42 34 ........+.2...B4
0x0050 2E 30 31 45 00 00 30 32 2E 33 37 41 20 20 00 00 .01E..02.37A ..
0x0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x0070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FC 02 ..............ü.
0x0080 00 00 00 00 EB 06 5D 1D 55 00 48 03 00 00 4D 04 ....ë.].U.H...M.
0x0090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x00A0 00 00 00 00 08 01 20 00 00 00 C4 09 F4 01 00 00 ...... ...Ä.ô...
0x00B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x00C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x00D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x00E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x00F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x0100 53 39 42 59 4C 4D 50 46 4D 35 48 55 43 32 52 46 S9BYLMPFM5HUC2RF
0x0110 56 46 4B 31 55 46 00 20 20 20 20 20 20 20 20 20 VFK1UF.
0x0120 20 20 20 20 30 39 2D 32 32 2D 32 30 30 37 00 00 09-22-2007..
0x0130 30 39 2D 32 32 2D 32 30 30 37 00 00 00 00 00 00 09-22-2007......
(0x0140 → 0x01F0: 0x00)
```

Página de configuración 0x05

```
0x0000 02 00 15 00 6F 59 1C 1D 6F 59 1C 1D 6F 59 1C 1D ....oY..oY..oY..
0x0010 7F 63 38 1D C0 FF FF FF FF FF FF 00 00 E0 FF □c8.Àÿÿÿÿÿÿ..àÿ
0x0020 00 00 FF FF FF FF 00 00 D0 02 D0 02 D0 02 D0 02 ..ÿÿÿÿ..Ð.Ð.Ð.Ð.
0x0030 D0 02 D0 02 00 00 00 00 FF 19 00 00 00 00 00 00 Ð.Ð.....ÿ.......
0x0040 00 00 5B C4 CF 02 00 00 9E 04 9E 04 9E 04 9E 04 ..[ÄÏ...ž.ž.ž.ž.
0x0050 9E 04 9E 04 00 1A 00 00 DF 34 00 00 5C C4 CF 02 ž.ž.....ß4..\ÄÏ.
0x0060 00 00 95 B8 B0 05 00 00 92 04 92 04 92 04 92 04 ..•¸°...'.'.'.'.
0x0070 92 04 92 04 E0 34 00 00 7F 47 00 00 96 B8 B0 05 '.'.à4..□G..-¸°.
0x0080 00 00 87 98 A7 07 00 00 80 04 80 04 80 04 80 04 ..‡˜§...€.€.€.€.
```

(0x0090 → 0x03FF): Información no relevante para el propósito de este proyecto

Arquitectura Marvell ROYL

Modelo de disco duro utilizado	WDC WD1600AAJS-60M0A0
SPT en área de servicio*	1311 [0x051F]

Cilindros en el área de servicio*	170 [0x00AA]
RPM del disco duro*	7200 [0x1C20]
Radios de servo (servo wedges)*	276 [0x0114]
Zonas*	21 [0x0015]
Platos instalados en el disco duro*	1 [0x01]
Cabezales físicos de lectura/escritura*	1 [0x01]
Cabezales virtuales (lógicos) de lectura/escritura*	1 [0x01]
Máscara de cabezales activos	0b00000001 [0x01]

Tabla 20: Ubicación de parámetros de interés en las páginas de configuración en discos duros WD familia Marvell ROYL.

* Información reportada por la herramienta utilizada.

Página de configuración 0x01

```
0x0000 02 00 30 31 2E 43 53 45 00 00 30 33 2E 30 39 00 ..01.CSE..03.09.
0x0010 00 00 30 31 2E 43 53 45 00 00 CB 16 01 01 01 01 ..01.CSE..Ë.....
0x0020 00 15 14 01 AA 00 20 1C 00 00 00 00 28 4D 82 06 ....ª. .....(M,.
0x0030 00 02 00 00 A1 46 00 00 A1 46 58 00 20 00 00 00 ....¡F..¡FX. ...
0x0040 05 00 00 00 00 00 01 16 2B 00 37 0A 00 01 41 31 ........+.7...A1
0x0050 2E 30 30 32 00 00 30 39 31 34 33 34 34 53 B6 1B .002..0914344S¶.
0x0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x0070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 B9 05 ..............¹.
0x0080 00 00 00 00 D8 7A AF 12 00 00 8A 0C 00 00 B2 04 ....Øz¯...Š...².
0x0090 00 04 00 00 9D 01 00 00 00 00 00 00 11 00 22 00 ....□.........".
0x00A0 04 2A 00 00 0C 01 10 00 00 00 C4 09 F4 01 34 21 .*........Ä.ô.4!
0x00B0 34 21 1E 33 00 00 08 00 30 30 4D 30 30 30 36 30 4!.3....00M00060
0x00C0 01 00 18 00 01 00 00 00 00 00 00 00 00 00 00 00 ................
0x00D0 00 00 57 44 43 20 57 44 31 36 30 30 41 41 4A 53 ..WDC WD1600AAJS
0x00E0 2D 36 30 4D 30 41 30 20 20 20 20 20 20 20 20 20 -60M0A0
0x00F0 20 20 20 20 20 20 20 20 20 20 00 00 00 00 00 00 ......
0x0100 53 7C 42 7C 4C 51 50 38 4D 4B 48 37 43 59 52 4D S|B|LQP8MKH7CYRM
0x0110 56 4E 4B 30 55 46 00 20 20 20 20 20 20 20 20 20 VNK0UF.
0x0120 20 20 20 20 30 31 2D 31 36 2D 32 30 30 39 00 00 01-16-2009..
0x0130 00 00 00 00 00 00 00 00 00 00 00 00 00 10 00 57 44 ..............WD
0x0140 2D 57 43 41 56 33 30 36 32 30 32 36 36 00 00 00 -WCAV30620266...
(0x0150 → 0x01F0: 0x00)
```

Página de configuración 0x05

```
0x0000 00 10 15 00 01 00 AF 9E A1 12 00 00 AF 9E A1 12 ......¯ž¡...¯ž¡.
0x0010 00 00 AF 9E A1 12 00 00 B0 9E A1 12 00 00 AF 9E ..¯ž¡...°ž¡...¯ž
0x0020 D1 12 00 00 95 6B D2 12 00 00 D8 7A DF 12 00 00 Ñ...•kÒ...Øzß...
0x0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x0050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x0060 00 00 00 00 00 00 00 00 00 00 00 00 1F 05 00 00 ................
0x0070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x0080 00 00 79 8B A3 01 00 00 4E 07 00 00 00 00 00 00 ..y‹£...N.......
0x0090 00 00 00 00 00 00 00 00 7A 8B A3 01 00 00 6F 43 ........z‹£...oC
0x00A0 1B 03 00 00 30 07 00 00 00 00 00 00 00 00 00 00 ....0...........
0x00B0 00 00 00 00 70 43 1B 03 00 00 9B 92 05 05 00 00 ....pC....›'....
0x00C0 02 07 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x00D0 9C 92 05 05 00 00 B1 0D FA 05 00 00 E6 06 00 00 œ'....±.ú...æ...
0x00E0 00 00 00 00 00 00 00 00 00 00 00 00 B2 0D FA 05 ............².ú.
0x00F0 00 00 8A 09 4E 07 00 00 BD 06 00 00 00 00 00 00 ..Š.N...½.......
0x0100 00 00 00 00 00 00 00 00 8B 09 4E 07 00 00 61 0E ........‹.N...a.
0x0110 78 08 00 00 8F 06 00 00 00 00 00 00 00 00 00 00 x...□...........
0x0120 00 00 00 00 62 0E 78 08 00 00 29 72 AE 09 00 00 ....b.x...)r®...
0x0130 78 06 00 00 00 00 00 00 00 00 00 00 00 00 00 00 x...............
0x0140 2A 72 AE 09 00 00 BA 9C C7 0A 00 00 33 06 00 00 *r®...ºœÇ...3...
0x0150 00 00 00 00 00 00 00 00 00 00 00 00 BB 9C C7 0A ............»œÇ.
0x0160 00 00 56 2C DD 0B 00 00 FA 05 00 00 00 00 00 00 ..V,Ý...ú.......
0x0170 00 00 00 00 00 00 00 00 57 2C DD 0B 00 00 C8 EB ........W,Ý...Èë
0x0180 EA 0C 00 00 C0 05 00 00 00 00 00 00 00 00 00 00 ê...À...........
0x0190 00 00 00 00 C9 EB EA 0C 00 00 50 9A A4 0D 00 00 ....Éëê...Pš¤...
0x01A0 8B 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ‹...............
0x01B0 51 9A A4 0D 00 00 E0 15 8F 0E 00 00 64 05 00 00 Qš¤...à.□...d...
0x01C0 00 00 00 00 00 00 00 00 00 00 00 00 E1 15 8F 0E ............á.□.
0x01D0 00 00 F3 73 35 0F 00 00 1F 05 00 00 00 00 00 00 ..ós5...........
0x01E0 00 00 00 00 00 00 00 00 F4 73 35 0F 00 00 3F A0 ........ôs5...?
0x01F0 E0 0F 00 00 E6 04 00 00 00 00 00 00 00 00 00 00 à...æ...........
0x0200 00 00 00 00 40 A0 E0 0F 00 00 32 29 99 10 00 00 ....@ à...2)™...
0x0210 AC 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ¬...............
0x0220 33 29 99 10 00 00 E8 0F 10 11 00 00 7E 04 00 00 3)™...è.....~...
0x0230 00 00 00 00 00 00 00 00 00 00 00 00 E9 0F 10 11 ............é...
0x0240 00 00 80 4E AB 11 00 00 50 04 00 00 00 00 00 00 ..€N«...P.......
0x0250 00 00 00 00 00 00 00 00 81 4E AB 11 00 00 8E C6 ........□N«...ŽÆ
0x0260 08 12 00 00 12 04 00 00 00 00 00 00 00 00 00 00 ................
0x0270 00 00 00 00 8F C6 08 12 00 00 76 7E 62 12 00 00 ....□Æ....v~b...
```

```
0x0280 E8 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 è...............
0x0290 77 7E 62 12 00 00 AF 9E A1 12 00 00 C6 03 00 00 w~b...¯ž¡...Æ...
(0x02A0 → 0x11EF: 0x00)
0x11F0 00 00 00 00 00 00 00 00 00 00 00 00 84 00 00 00 ............„…
```

Arquitectura Marvell ROYL 20B

Modelo de disco duro utilizado	WDC WD5000AZRX-00L4HB0
SPT en área de servicio*	1980 [0x07BC]
Cilindros en el área de servicio*	256 [0x0100]
Platos instalados en el disco duro*	1 [0x01]
Cabezales físicos de lectura/escritura*	2 [0x02]
Cabezales virtuales (lógicos) de lectura/escritura*	1 [0x01]
Máscara de cabezales activos	0b00000001 [0x01]

Tabla 21: Ubicación de parámetros de interés en las páginas de configuración en discos duros WD familia Marvell ROYL 20B

* Información reportada por la herramienta utilizada.

Página de configuración 0x01

```
0x0000 02 00 30 31 2E 33 34 47 00 00 32 32 2E 33 30 00 ..01.34G..22.30.
0x0010 00 00 30 31 2E 33 34 47 00 00 FE 16 01 02 01 01 ..01.34G..þ.....
0x0020 00 3D 00 02 00 01 18 15 00 00 00 00 37 4D 82 05 .=..........7M,.
0x0030 00 02 00 00 C0 A5 00 00 C0 A5 E9 00 21 00 00 00 ....À¥..À¥é.!...
0x0040 03 00 00 00 25 80 00 00 00 00 00 0C 00 01 45 58 ....%€........EX
0x0050 2E 30 30 30 00 00 47 2E 30 30 30 32 35 41 B6 FD .000..G.00025A¶ý
0x0060 C7 4D 23 02 00 00 00 00 00 00 00 00 00 00 00 00 ÇM#.............
0x0070 00 00 00 00 00 00 20 00 00 00 00 00 00 00 00 00 ...... .........
0x0080 00 00 00 00 D0 0C 43 3A 00 00 00 00 00 05 0A 05 ....Ð.C:........
0x0090 00 02 00 00 9D A0 40 00 00 00 00 00 00 00 00 00 ....□ @.........
0x00A0 00 00 00 00 10 01 02 00 00 00 C4 09 F4 01 00 00 ..........Ä.ô...
0x00B0 00 00 0C E1 01 00 40 00 FF FF FF FF FF FF FF FF ...á..@.ÿÿÿÿÿÿÿÿ
0x00C0 00 00 00 00 00 00 00 00 80 00 00 00 00 00 00 00 ........€.......
0x00D0 00 00 57 44 43 20 52 4F 4D 20 4D 4F 44 45 4C 2D ..WDC ROM MODEL-
0x00E0 54 52 41 49 4C 58 4C 53 2D 00 20 20 20 20 20 20 TRAILXLS-.
0x00F0 20 20 20 20 20 20 20 20 20 20 00 00 00 00 00 00 ......
```

```
0x0100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x0110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x0120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x0130 00 00 00 00 00 00 00 00 00 00 00 00 10 00 57 44 ..............WD
0x0140 43 2D 52 4F 4D 20 53 4E 23 20 58 59 5A 2D 2D 2D C-ROM SN# XYZ---
0x0150 2D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 -................
0x0160 00 00 00 00 00 00 00 00 00 00 00 00 07 02 8F 07 ..............□.
0x0170 1E 00 01 00 08 00 00 00 00 80 67 00 00 00 00 00 .........€g.....
0x0180 00 00 00 00 00 00 00 00 00 00 00 00 00 00 AB 0A ..............«.
0x0190 00 00 00 00 40 00 00 00 00 00 00 00 00 00 00 00 ....@...........
0x01A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x01B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x01C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x01D0 00 02 00 02 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x01E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x01F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
```

Página de configuración 0x05

```
0x0000 00 10 43 01 06 00 AF 88 E0 E8 00 00 AF 88 E0 E8 ..C...¯ˆàè..¯ˆàè
0x0010 00 00 AF 88 E0 E8 00 00 A0 00 DF E8 00 00 AF 88 ..¯ˆàè.. .ßè..¯ˆ
0x0020 35 E9 00 00 00 00 00 00 00 00 AF 88 35 E9 00 00 5é........¯ˆ5é..
0x0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x0050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x0060 00 00 00 00 00 00 00 00 00 00 00 00 BC 07 BC 07 ............¼.¼.
0x0070 BC 07 BC 07 BC 07 BC 07 00 00 00 00 00 00 00 00 ¼.¼.¼.¼.........
0x0080 00 00 EF 91 E6 06 00 00 7E 01 8C 01 77 01 A8 01 ..ï'æ...~.Œ.w.¨.
```

(0x0090 → 0x35FF): Información no relevante para el propósito de este proyecto

Anexo 2

Cálculo del tamaño del firmware en área de servicio dado el directorio de módulos

Este cálculo se ha realizado apoyado en un pequeño programa escrito en Java, desarrollado para este proyecto, que recorre el directorio de módulos entero, y suma el tamaño de cada uno de los módulos listados. No es un programa genérico, y se apoya en haber identificado visualmente y a priori que el tamaño de cada entrada de módulo tiene tamaño 0x1A bytes (para un caso concreto). Se adjunta un extracto del mismo (siguiente página):

```
public static void main(String [] args) throws Exception {
   int totalSectors = 0;
   RandomAccessFile f = new RandomAccessFile("/tmp/mod_01.bin", "r");
   byte [] buffer = new byte[0x1A];
   int read = 0;
   int iter = 0;
   f.seek(0x4C);
   while ((read = f.read(buffer)) != -1) {
       if (read == 0x1A) { //Hemos leído una fila entera de la tabla
           String _zero = hex(buffer[0]);
           String _lo = hex(buffer[4]);
           String _hi = hex(buffer[5]);
           System.out.println("Read #" + iter + ": " + _zero);
           if ("1a".equalsIgnoreCase(_zero)) {
               int size = Integer.parseInt(_hi + _lo, 16);
               totalSectors += size;
               System.out.println("Size="+size+" ["+_hi+ ","+_lo+"]");
           }
       }
       iter++;
   }
   f.close();
   System.out.println("\n\n" + "Total sectors = " + totalSectors);
   System.out.println("Total size = " + totalSectors * 0.5f + " kB");
}
```

Anexo 3

Modelos de discos duros Western Digital por familia

Los modelos de los discos duros Western Digital tienen un código que identifica la familia de la siguiente manera [79]:

WD1234ABCD - **XXYY(Y)ZZ**

La parte que queda a la derecha del guión es solo para uso interno de la compañía Western Digital [80]; no obstante existen documentos de compañías especializadas en recuperación de información de discos duros dañados que establecen una relación entre los nombres de familia de los discos duros y este código [81].

La familia queda identificada por **YY(Y)**, pudiendo ser éste un código de 2 o de 3 caracteres.

Ejemplo: WD2500JS - 22NCB1

El código de familia es **NC**; según la tabla queda identificado como un disco duro WD Hawk familia WD Marvell.

Nombre (uso interno)	Código(s) identificable(s)	Familia
Aquarius	KZ, LA, RM, RN	Marvell
Buccaneer	KE, KF, KG, KM	Marvell
Hawk	MH, MJ, MK, ML, MV, MW, MY, MZ, NC, ND, NE, NF, NG, NH, NJ, NK, NT, NU, NV, NW, NY, NZ, PA, PB, RP	Marvell
Hawk2	SG, TG	Marvell

Mammoth	EY, EZ, FA, FC, FJ, FM, HE, HF, JE, JS, JT, JY	Marvell
Orion	KV, KW	Marvell
Raider	PC, PD, PE, PF, PG	Marvell
Sabre	JH, JJ, JK, JL, JM, JN, JP, JR, JU, KS, LN, MG	Marvell
Scorpio	HC, HD, KH	Marvell
Starling	RD, RE, RF, RJ, RK, RL	Marvell
Unicorn	LR, LS, LT, LU, MR, MS, MT, MU	Marvell
Zeus	MN, MP, VJ	Marvell
Aries	RR, RS, RT, RU, RV, RW	Marvell ROYL
Atlantis	A7, A8, A9, B0, B1, B2, C1, C2, C3, D2, E7	Marvell ROYL
Cougar	WY, WZ	Marvell ROYL
Cypress	G0, G1, G2, G3, G4, K0, J9	Marvell ROYL
Denali	UZ, VA, VB, VC, VD, VE, ZD, ZC	Marvell ROYL
Hulk	C7, ZJ, ZK, ZL, ZP, ZR	Marvell ROYL
Jupiter	RB, RC, TU	Marvell ROYL
Kermit	D6, D7, E0, E1	Marvell ROYL
Lynx	UW, UY	Marvell ROYL
Mars	A0, A1, A2, A3, A4, A5, A6, D4, E8, E9, F5, F6, F7, J7, J8, K1	
Mercury	ZA, ZB	Marvell ROYL
Saturn	F3, F4	Marvell ROYL
Sequoia	PN, PR, PS, PT, PU, PV, SV, SW, SY, SZ	Marvell ROYL
Sequoia PMR	C0, D1, VZ, WA, WB, WM, WN, WP, WR, WS, WT	Marvell ROYL

Spider	D0, ZS, ZT, ZU, ZV, ZW, ZY, ZZ	Marvell ROYL
STG Twin Lakes	VK, VL, VM, VN, VP, VR, VS, VT, VU	Marvell ROYL
Tornado	RY, RZ, SB, SC, SD, SE, TA, TB, TC, TV, TW, TY, TZ, UA, UB, UC, UD, UE, UH	Marvell ROYL
Tornado 2D	B9, C9, VV, VW, VY, WC, WD, WE	Marvell ROYL
Tornado 2PMR	WF, WG, WH, WJ, WK, WL	Marvell ROYL
Tornado 2R	C8, YE, YF, YG, YH, YJ, YK, YL, YM, YN, YP, YR, YS, YT, YU, YV, YW, YY, YZ	Marvell ROYL
Tornado 3D	TH, TJ, TK, TL, TM, TN, TP, TR, TS, UF, UG, UJ, UK, UL, UM, UN, UP, UH	Marvell ROYL
DF4PL RE	W3D	Marvell ROYL 20B
DragFly1	M7, M8, N7, S9, U7	Marvell ROYL 20B
DragFly2	M2, M3, M4, M5, P6, U8	Marvell ROYL 20B
DragFly3	P8, P9, R0, Z5	Marvell ROYL 20B
DragFly4	H5, H7, P3, R6, S1, S2, S8, T3, U1, U2	Marvell ROYL 20B
Dragon	J99	Marvell ROYL 20B
Espirit	HPJ, PK4	Marvell ROYL 20B
Firebird	A1Y, S5X	Marvell ROYL 20B
FBLite	AJG, JC3	Marvell ROYL 20B
Helios	TK7, U4M, ZSM	Marvell ROYL 20B
Hubble LT	V0T, FMC, VED, G33	Marvell ROYL 20B
Mariner	A05, A06, A1C, A0L, A0R	Marvell ROYL 20B
Midori	L0, L1, L2, K9	Marvell ROYL 20B
Pinclite	L6, L7, L8, L9, M0, M1, N9, J2, J3, J5, R1	Marvell ROYL 20B

Pinnacle	B3, B4, B5, B6, B7, E2, F0, F1, F2, H4	Marvell ROYL 20B
Sadle G6	MVW	Marvell ROYL 20B
Shasta K6	A02, A1J, A1N, A28	Marvell ROYL 20B
Shrek	Z2T	Marvell ROYL 20B
Shrek LT	AV3, W68	Marvell ROYL 20B
Tahoe 2D	E3, Z8	Marvell ROYL 20B
Tahoe LT	1C, YZC, UU3	Marvell ROYL 20B
Tahoe	M9, N0, N1, N2, N3, V0, V1	Marvell ROYL 20B
Tressels	RKK, ZF5	Marvell ROYL 20B
Venus	A04, A1P, F9	Marvell ROYL 20B

Tabla 22: Códigos de familia de discos duros por arquitectura

Anexo 4

Compilación y uso del controlador

El controlador consta de un único fichero, *afbd.c*, que contiene todo el código C necesario para funcionar (salvo las librerías estándar y de Linux). Para utilizarlo en un sistema Linux con un disco duro compatible, es preciso conectarlo a un puerto ATA (S-ATA o P-ATA) disponible, averiguar la dirección del puerto físico, e indicar al controlador, en el código fuente, que haga uso de la misma:

```
static int baseport = 0xb080;
```
[20]

Una vez indicada la dirección del puerto base donde se encuentra el disco duro, podemos compilarlo:

```
# make -C /lib/modules/`uname -r`/build M=`pwd` modules
```

El comando make hace uso del siguiente fichero de *Makefile*:

```
obj-m := afbd.o
KDIR := /lib/modules/$(shell uname -r)/build
PWD := $(shell pwd)
default:
    $(MAKE) -C $(KDIR) SUBDIRS=$(PWD) modules
```

Nótese que el comando recibe como parámetro una ruta que contiene el resultado de ejecutar `uname -r` en Linux, que devuelve el nombre del *kernel* de Linux instalado, por lo que es necesario tener las cabeceras del *kernel* instaladas en la ruta adecuada. Las cabeceras del *kernel* que deben usarse en la compilación del módulo deben ser compatibles con la versión 2.6 en cuanto a la cabecera *blkdev.h* se refiere; esto es versiones iguales o superiores a la 2.6 e inferiores a la 3.16.

[20] Obtenido usando los comandos en Linux *lspci -v, lspci -D,* y *ls -l /sys/bus/pci/devices/0000\:00\:1f.5/host4/target4\:0\:0/4\:0\:0\:0/block/*

Si la compilación es satisfactoria, obtenemos como resultado el fichero ***afdb.ko***, que es el módulo (controlador) usable.

Para usarlo, se ejecuta el siguiente comando con privilegios de administrador:

```
# insmod afbd.ko
```

Tras la inserción del módulo con el anterior comando, el controlador inicia su secuencia de arranque, trata de detectar el disco duro conectado en el puerto indicado, y en caso de que no encuentre problemas o incompatibilidades, presenta una unidad al sistema operativo, con tamaño igual al espacio disponible calculado.

Referencias

[1] National Criminal Justice Reference Service. *Best Practices For Seizing Electronic Evidence v.3: A Pocket Guide for First Responders*. Extraído de <https://www.ncjrs.gov/App/publications/Abstract.aspx?id=239359>. [Consultado en septiembre de 2015].

[2] O'Grady Rueda, D. (2014). *Diseño de un driver de disco duro para sistemas de ficheros indetectables*. Conferencia *No Con Name 2014*. Barcelona, 31 octubre 2014.

[3] ANSI INCITS 397-2005 standard. *ATA/ATAPI-7*, vol. 1-3.

[4] Samsung Electronics. *Samsung F2EG Product Guide*, p. 2. Extraído de <http://www.seagate.com/files/www-content/support-content/documentation/samsung/tech-specs/eco_greenf2.pdf>. [Consultado en septiembre 2015].

[5] ANSI INCITS 397-2005 standard. *ATA/ATAPI-7*, rev. 4b (T13/1532D), vol. 3, cl. 13-19.

[6] ANSI INCITS 397-2005 standard. *ATA/ATAPI-7*, rev. 4b (T13/1532D), vol. 3, cl. 20.

[7] ANSI INCITS 397-2005 standard. *ATA/ATAPI-7*, rev. 4b (T13/1532D), vol. 1, cl. 6.12.

[8] BS ISO/IEC 27037:2012. *Reproducibility*, p. 7.

[9] Extraído de la página web del fabricante Atola <http://www.atola.com/products/insight/>. [Consultado en septiembre 2015].

[10] Extraído de la página web del fabricante ACE Laboratory <http://www.acelaboratory.com/pc3000.udma.php>. [Consultado en septiembre 2015].

[11] Extraído de la página web del fabricante Salvation Data <http://mail.salvationdata.com/data-recovery-equipment/hd-doctor.htm>. [Consultado en septiembre 2015].

[12] Extraído de la página web del fabricante MRT Lab <http://en.mrtlab.com/mrt-pro>. [Consultado en septiembre 2015].

[13] HDDScan. *HDD From Inside. Main Parts.* Extraído de <http://hddscan.com/doc/HDD_from_inside.html>. [Consultado en septiembre 2015].

[14] ACE Laboratory. *PC3000 UDMA, How to use*. Extraído de la página del fabricante <http://www.acelaboratory.com/pc3000.udma.php>. [Consultado en septiembre 2015].

[15] Malenkovich, Serge. Kaspersky Blog. *Indestructible Malware by Equation cyberspies is out there - but don't panic (yet).* 17 febrero 2015. Extraído de <https://blog.kaspersky.com/equation-hdd-malware/7623/>. [Consultado en septiembre 2015].

[16] Sobey, Charles H. ChannelScience. *Computational Methods of Signal Decoding for Recovering Recorded Data*, NASA/IEEE MSST 2004 conference, p. 22. Extraído de <http://storageconference.us/2004/Presentations/WIP/36-04.pdf>. [Consultado en septiembre 2015].

[17] Hitachi Ultrastar 15K147 SCSI Hard Disk Drive Specification, v. 3.1 noviembre 2006, p. 19. (Falta de referencias más genéricas debido a que esta característica no forma parte del estándar).

[18] Sobey, Charles H. IEEE Senior Member, Orto L. et al. *Drive-Independent Data Recovery: The Current State-of-the-Art*. Preprint, D5, p. 3.

[19] A·FF Laboratory. *A·FF Repair Station. What is Firmware Area?* Extraído de <http://www.hdd-tools.com/products/rrs/>. [Consultado en septiembre 2015].

[20] Sobey, Charles H. ChannelScience. *Computational Methods of Signal Decoding for Recovering Recorded Data*, NASA/IEEE MSST 2004 conference. Extraído de <http://storageconference.us/2004/Presentations/WIP/36-04.pdf>. [Consultado en septiembre 2015].

[21] Sobey, Charles H. IEEE Senior Member, Orto L. et al. *Drive-Independent Data Recovery: The Current State-of-the-Art*. Preprint, D5, p. 3. "For some drive models, the system area contains only a small amount of information, such as a unique drive serial number (...)"

[22] ACE Laboratory. *Western Digital "Spartan", "Protege", "Caviar" Generation electronics Arch-V, Arch-VI "PCWD-DA", "PCWD-EB", "PCWD-ABJ", "PCWD-CB2"*. pp. 13-14, *Структура информации в Flash ПЗУ, в семействах WD Caviar, Protégé* (estructura de la memoria Flash/ROM en las familias de Western Digital Caviar y Protégé). Extraído de <http://www.acelab.ru/dep.pc/doc.pc3000dos/060713.001/pcwd_abj_15_03.pdf>. [Consultado en septiembre de 2015].

[23] Sobey, Charles H. IEEE Senior Member, Orto L. et al. *Drive-Independent Data Recovery: The Current State-of-the-Art*. Preprint, D5, p. 3. "(...) Often they are located at the extreme outer diameter (OD); however, they have been found at other radii as well."

[24] Microsoft Corporation. *MS-DOS Partitioning Summary*, rev. 3.0. Article ID 69912.

[25] Microsoft Corporation. *Using GPT Drives*. Actualizado el 19 de octubre de 2010. Extraído de <https://msdn.microsoft.com/en-us/library/windows/hardware/dn653580(v=vs.85).aspx>. [Consultado en octubre de 2015].

[26] ACE Laboratory. *Western Digital "Spartan", "Protege", "Caviar" Generation electronics Arch-V, Arch-VI "PCWD-DA", "PCWD-EB", "PCWD-ABJ", "PCWD-CB2"*. Sección 5.3, *Структура служебной информации загружаемой части (DISK F/W)*

(Estructura del área de servicio del firmware del disco duro). Extraído de <http://www.acelab.ru/dep.pc/doc.pc3000dos/060713.001/pcwd_abj_15_03.pdf>. [Consultado en septiembre de 2015].

[27] Sobey, Charles H. IEEE Senior Member, Orto L. et al. *Drive-Independent Data Recovery: The Current State-of-the-Art*. Preprint, D5, p. 3. "(...) if a module of this information is corrupt in one copy, a good copy of the module (...)".

[28] ANSI INCITS 397-2005 standard. *ATA/ATAPI-7*, vol. 1, rev. 4b. *General Operational Requirements*. p. 15.

[29] Sobey, Charles H. IEEE Senior Member, Orto L. et al. *Drive-Independent Data Recovery: The Current State-of-the-Art*. Preprint, D5, p. 3. "(...) often a 'translator' that converts between logical and physical block addresses including the effects of head and track skewing, S.M.A.R.T. data, and a (possibly encrypted) drive password."

[30] ANSI INCITS 397-2005 standard. *ATA/ATAPI-7*, vol. 1, rev. 4b. p. 22.

[31] Extraído de <http://www.wikinvest.com/stock/Western_Digital_(WDC)>. [Consultado en septiembre de 2015].

[32] Extraido de <http://www.digitimes.com/news/a20100513PR201.html>. [Consultado en septiembre de 2015].

[33] Extraído de <http://www.marvell.com/company/news/pressDetail.do?releaseID=2236> [Consultado en septiembre de 2015].

[34] ACE Laboratory. *Western Digital "Spartan", "Protege", "Caviar" Generation electronics Arch-V, Arch-VI "PCWD-DA", "PCWD-EB", "PCWD-ABJ", "PCWD-CB2"*. Secciones 5.3 y 5.4. Extraído de <http://www.acelab.ru/dep.pc/doc.pc3000dos/060713.001/pcwd_abj_15_03.pdf>. [Consultado en septiembre de 2015].

[35] Dolphin Datalab. *HDD Repair - WD HDD Repair Head Depopping Options and Steps*. 29 junio 2012. Extraído de <http://www.dolphindatalab.com/hdd-repair-wd-hdd-head-depopping-options-and-steps/>. [Consultado en septiembre 2015].

[36] Sobey, Charles H. IEEE Senior Member, Orto L. et al. *Drive-Independent Data Recovery: The Current State-of-the-Art*. Preprint, D5, p. 3. "(...) many manufacturers store multiple copies of the system area information, often one (or more) per surface."

[37] ACE Laboratory. *Western Digital "Spartan", "Protege", "Caviar" Generation electronics Arch-V, Arch-VI "PCWD-DA", "PCWD-EB", "PCWD-ABJ", "PCWD-CB2"*. Sección 5.3. Extraído de <http://www.acelab.ru/dep.pc/doc.pc3000dos/060713.001/pcwd_abj_15_03.pdf>. [Consultado en septiembre de 2015].

[38] Rubstov, A. HDDScan. *HDD from inside: Tracks and zones. How hard can it be?*. Agosto 2009. Extraído de <http://hddscan.com/doc/HDD_Tracks_and_Zones.html>. [Consultado en septiembre de 2015].

[39] ACE Laboratory. *WDC Marvell Utility Manual* (22 agosto 2013). s. 4.9 *DIR Editing*, p. 41.

[40] Catálogo de productos con especificaciones técnicas extraído de <http://www.wdc.com/en/products/legacy/Legacy.asp?Model=WD600BB>. [Consultado en septiembre de 2105].

[41] idle3-tools. *WD Specific ATA commands. VSC Enable*. Extraído de <http://idle3-tools.sourceforge.net/>. [Consultado en septiembre de 2015].

[42] The Linux Documentation Project. *2. Using I/O ports in C programs. Accessing the ports*. Extraído de <http://tldp.org/HOWTO/IO-Port-Programming-2.html>. [Consultado en septiembre de 2015].

[43] idle3-tools. *WD Specific ATA commands. VSC Disable*. Extraído de <http://idle3-tools.sourceforge.net/>. [Consultado en septiembre de 2015].

[44] ANSI INCITS 397-2005 standard. *ATA/ATAPI-7*, vol. 1, rev. 4b. s. 6.54.8. *SMART WRITE LOG*. p. 301.

[45] ANSI INCITS 397-2005 standard. *ATA/ATAPI-7*, vol. 1, rev. 4b. s. 6.54.6. Table 52, *Log address definition*. p. 290.

[46] idle3-tools. *WD Specific ATA commands. VSC Send Write Key.* Extraído de <http://idle3-tools.sourceforge.net/>. [Consultado en septiembre de 2015].

[47] Berkman, A. Recover Information Technologies, LTD. *Hiding Data in Hard Drive's Service Areas*. Febrero 2013. Extraído de <http://www.recover.co.il/SA-cover/SA-cover.pdf> y de <http://www.recover.co.il/SA-cover/SA-cover-poc.c>. [Consultado en septiembre de 2015].

[48] Wikipedia. *Little Endian,* Extraído de <https://en.wikipedia.org/wiki/Endianness>. [Consultado en septiembre de 2015].

[49] Linux Programmer's Manual. *OUTB(2), void insw(unsigned short int port, void *addr, unsigned long int count)*. Extraído de <https://www.linux.com/learn/docs/man/outb2> [Consultado en octubre 2015].

[50] Linux Programmer's Manual. *OUTB(2). void outsw(unsigned short int port, void *addr, unsigned long int count)*. Extraído de <https://www.linux.com/learn/docs/man/outsw2> [Consultado en octubre 2015].

[51] Rubstov, A. HDDScan. *HDD from inside: Tracks and zones. How hard can it be?*. Agosto 2009. Extraído de <http://hddscan.com/doc/HDD_Tracks_and_Zones.html>. [Consultado en septiembre de 2015].

[52] Johnson, Michael K. Linux Journal. *An introduction to block device drivers*. Extraido de <http://www.linuxjournal.com/article/2890>: "(...) and the sectors are numbered consecutively, starting at 0. If the physical device is addressed by some means other than sectors, it is the responsibility of the request() procedure to translate". [Consultado en septiembre de 2015].

[53] Extraído de <http://malthus.mooo.com/viewtopic.php?t=1011>. [Consultado en septiembre de 2015].

[54] ACE Laboratory. *Архитектура WD Marvell ROYL-20B* (Arquitectura de los WD Marvell ROYL 20B). p. 1. Extraído de <http://www.osslab.org.tw/@api/deki/files/3965/> [Consultado en septiembre de 2015].

[55] Berkman, A. Recover Information Technologies, LTD. *Hiding Data in Hard Drive's Service Areas*. Febrero 2013. Extraído de <http://www.recover.co.il/SA-cover/SA-cover.pdf> y de <http://www.recover.co.il/SA-cover/SA-cover-poc.c> [Consultado en septiembre de 2015].

[56] Microsoft. *What is a signed driver?* Extraído de <http://windows.microsoft.com/en-us/windows-vista/what-is-a-signed-driver> [Consultado en septiembre de 2015].

[57] Corbet, J. *et al*. *Linux Device Drivers, Third Edition*. (2005). Editorial O'Reilly. c. 16. *Block drivers*, p. 464-465.

[58] The Debian Project. Información sobre el proyecto Debian disponible en <https://www.debian.org/intro/about>. [Consultado en octubre de 2015].

[59] The GNU nano editor. Extraído de <http://www.nano-editor.org/dist/v2.2/faq.html#1.3>. [Consultado en octubre 2015].

[60] The Linux Documentation Project. *Makefile*. Extraído de <http://tldp.org/HOWTO/Software-Building-HOWTO-3.html>. [Consultado en octubre de 2015].

[61] Corbet, J. Eklektix, Inc. *A simple block driver*. (17 de noviembre, 2003). Extraído de <https://lwn.net/Articles/58719/> [Consultado en octubre de 2015].

[62] Corbet, J. *et al*. *Linux Device Drivers, Third Edition*. (2005). Editorial O'Reilly. c. 16. *Block drivers*, p. 465-484.

[63] The Linux Documentation Project. *Partitions Mass Storage Definitions-Naming*. c. 5. *Drive naming in Linux*. Extraído de <http://www.tldp.org/HOWTO/Partition-Mass-Storage-Definitions-Naming-HOWTO/x99.html>. [Consultado en octubre de 2015].

[64] Corbet, J. Eklektix, Inc. *A simple block driver*. (17 de noviembre, 2003). *The request method*. Extraído de <https://lwn.net/Articles/58719/> [Consultado en octubre de 2015].

[65] Baker, T., Wang, A. *Block Drivers*. CIS 4930 / COP 5641. p. 21. Extraído de <http://www.cs.fsu.edu/~baker/devices/notes/lecture_block.ppt>. [Consultado en octubre de 2015].

[66] Russell, R. IBM Corporation. *insmod(8)*. Extraído de <http://linux.die.net/man/8/insmod>. [Consultado en octubre de 2015].

[67] ANSI INCITS 397-2005 standard. *ATA/ATAPI-7*, rev. 4b (T13/1532D), vol. 1, cl. 6.17. *IDENTIFY DEVICE*. p. 114.

[68] ANSI INCITS 397-2005 standard. *ATA/ATAPI-7*, rev. 4b (T13/1532D), vol. 1, cl. 6.17. Table 16. *IDENTIFY DEVICE data*. p. 116-118.

[69] ANSI INCITS 397-2005 standard. *ATA/ATAPI-7*, rev. 4b (T13/1532D), vol. 1, cl. 3.2.9. *Byte ordering for data transfers*. p. 13.

[70] Wikipedia. *Logical Block Addressing. CHS Conversion*. Extraído de <https://en.wikipedia.org/wiki/Logical_block_addressing>. [Consultado en octubre de 2015].

[71] ANSI INCITS 397-2005 standard. *ATA/ATAPI-7*, rev. 4b (T13/1532D), vol. 1, cl. 5. *I/O register descriptions*. p. 51.

[72] ANSI INCITS 397-2005 standard. *ATA/ATAPI-7*, rev. 4b (T13/1532D), vol. 1, cl. 5.14. *Status register*. p. 61.

[73] Wikipedia. *Debugging*. Techinques. *Print debugging*. Extraído de <https://en.wikipedia.org/wiki/Debugging#Techniques>. [Consultado en octubre de 2015].

[74] Laub, John H. NIST. *Test Results for Digital Data Acquisition Tool: VOOM HardCopy 3P – Firmware Version 2-04*. Septiembre de 2012. Documento disponible en <https://ncjrs.gov/pdffiles1/nij/238995.pdf>. [Consultado en septiembre de 2015].

[75] LWN. *Linux and 4K disk sectors*. *Sector Structure*. Extraído de <https://lwn.net/Articles/322777/>. [Consultado en octubre de 2015].

[76] Ayuntamiento de Madrid. *Expediente de contratación*. Acreditación informática. Nº de expediente: 300/2010/01097. Extraído de <http://www.madrid.es/perfilContratante%5Ca404f3c33082d210VgnVCM2000000c205a0aRCRD%5C1302182127631/acreditacion1302182127631.pdf>. [Consultado en septiembre de 2015].

[77] Extraído de <http://lxr.free-electrons.com/diff/include/linux/blkdev.h?v=3.13;diffval=3.16;diffvar=v>. La estructura en C *struct request* pierde varios campos, como *char* buffer*, que usa el controlador. [Consultado en octubre de 2015].

[78] Micron Technology, Inc. *TN-29-42: Wear-Leveling Techniques in NAND Flash Devices*.

[79] Western Digital Company. *Model Numbers - WD Internal Drives*. Extraído de <http://www.wdc.com/wdproducts/library/other/2579-001028.pdf>. [Consultado en octubre de 2015].

[80] Western Digital Company. *Model Numbers - WD Internal Drives. Model Number Suffix*. Extraído de <http://www.wdc.com/wdproducts/library/other/2579-001028.pdf>. [Consultado en octubre de 2015].

[81] ACELab. PC3000. *Кпассификция семейств накопитепей WD* (*Clasificación de familias de WD*). p. 1. *Кодификция продукции* (*códigos de producto*). Extraído de

<http://www.acelab.ru/dep.pc/doc.pc3000dos/060713.001/WD-main-N-2.pdf>.
[Consultado en octubre de 2015].

Bibliografía

1. Corbet, J., Rubini, A., Kroah-Hartman, G. (2005). *Linux Device Drivers, Third Edition*. Editorial O'Reilly. ISBN 978-0-596-00590-0; ISBN 10: 0-596-00590-3